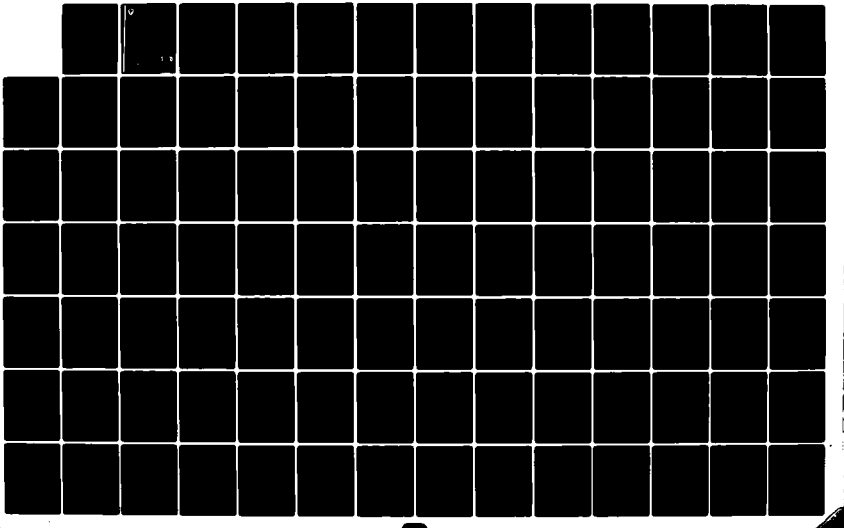
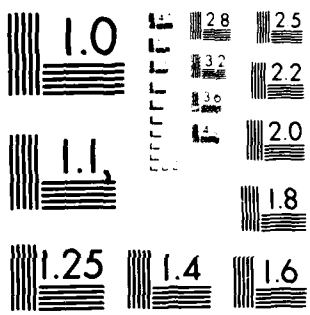
AD-A131 674 RESEARCH IN NET MANAGEMENT TECHNIQUES FOR TACTICAL DATA 1/2
NETWORKS(U) POLYTECHNIC INST OF NEW YORK BROOKLYN
R BOORSTYN ET AL. SEP 82 CECOM-80-0579-3
UNCLASSIFIED DAAK80-80-K-0579 F/G 17/2 NL





RESEARCH AND DEVELOPMENT TECHNICAL REPORT
CECOM-80-0579-3

Research in Net Management Techniques for Tactical Data Networks

Robert Boorstyn
Aaron Kershenbaum
POLYTECHNIC INSTITUTE OF NEW YORK
333 Jay Street
Brooklyn, New York 11201

September 1982

Semiannual Technical Report for Period September 1, 1981 to February 28, 1982

Approved for Public Release;
Distribution Unlimited.

Prepared for:
CENTER FOR COMMUNICATIONS SYSTEMS

CECOM

U S ARMY COMMUNICATIONS-ELECTRONICS COMMAND
FORT MONMOUTH, NEW JERSEY 07703

NOTICES

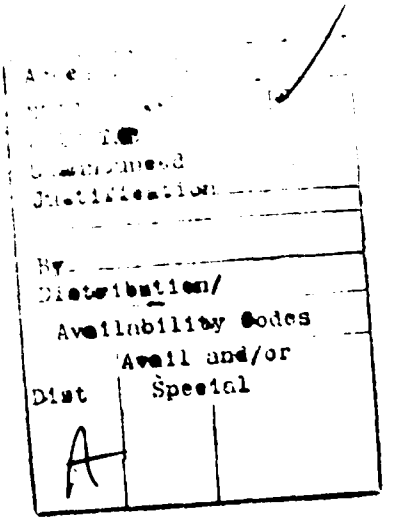
Disclaimers

The citation of trade names and names of manufacturers in this report is not to be construed as official Government indorsement or approval of commercial products or services referenced herein.

Disposition

Destroy this report when it is no longer needed. Do not return it to the originator.

TABLE OF CONTENTS



INTRODUCTION

Our research for this period is summarized in five parts below. In part A we extend our results on adaptive routing to further connect the two levels -- global and local. In part A.1 we present an efficient global routing algorithm which minimizes the maximum utilization in a link. This assures stability and generates required flows for the local level. It's simple objective function results in a very fast implementation. In part A.2 we reconsider the local level optimization but with output flow constraints. We show that there is a reasonable amount of control embedded in our priority rule. We also pose the problem as a Markov Decision Process, identify when deterministic policies are optimal, and show for a simple example that a threshhold policy is optimal.

In part B we present several results which generalize the analyses of random access in ALOHA, CSMA, and CSMA/CD to include arbitrary packet length distributions. In part C we extend our work on Multihop Packet Radio Networks to include non-exponential packet length distributions and present more examples, especially the throughput of a "hot spot."

In parts D and E we summarize some results on algorithms for Centralized Teleprocessing Network Design and on augmenting paths for Combinatorial Optimization Problems.

RESEARCH SUMMARIES

A. Studies in Adaptive Routing

A.1. Stable Routing Patterns

As part of a general investigation in the area of dynamic routing in computer communication networks we consider the problem of finding stable global routing patterns. Specifically, we have proposed [1] a two-level routing procedure where the lower (local) level adapts dynamically to instantaneous variations in the congestion of the network in the immediate vicinity of each node and the higher (global) level ensures stability by keeping the average load across the entire network in some sense globally balanced. We now consider this latter problem.

We consider as a measure of global balance the utilization of the most heavily utilized network element (node or link) and seek to minimize this quantity. For simplicity, we will speak only of link utilizations. (Node utilizations can be included in a straightforward manner.) Thus, we are given a network containing N nodes and L (directed) links. Each link, l, has a capacity C_l. There are (directed) requirements r_{ij} between nodes i and j. Each r_{ij} is satisfied by routing it on one or more paths $P^{(K)}_{ij}$ from i to j. (In the two level adaptive routing scheme, these paths (or links within them) will be the alternatives open to each requirement.) A routing pattern is defined by the paths $P^{(K)}_{ij}$ and the fraction, $f^{(K)}_{ij}$, of r_{ij} using each $P^{(K)}_{ij}$. The utilization of each link is equal to the total flow (sum of fractions of requirements) on the link divided by its capacity.

The maximally utilized link is in a sense the most vulnerable part of the network and the most likely cause for the dynamic routing mechanism to break down (e.g. loop) due to congestion. By minimizing the

utilization of the maximally utilized link we seek to minimize the chance of congestion leading to such failure. It should be noted that we are dealing with the global level of the routing procedure here and as such consider only long term average utilizations, not instantaneous measures. The local level of the routing procedure concerns itself with making decisions instantaneously on the basis of the local state of the network in the vicinity of a node. Even within the constraints of a given $P_{ij}^{(K)}$ and $f_{ij}^{(K)}$, defined by the global strategy, the local level has considerable flexibility in choosing when to use each route and as such can obtain substantial reductions in delay when compared with static routing policies.

We now turn to the problem of actually finding the optimal global routing pattern as defined above. The technique resembles the Flow Deviation Method of Cantor and Gerla [2] and will be described in similar terms. Cantor and Gerla sought to minimize the average delay whereas we seek to minimize the maximum utilization of a link. Both functions are convex functions over a convex region and as such, the same type of procedure can be proven to yield an optimal routing pattern. The proof is given in [2].

Our function is only piecewise differentiable and as such, the gradient search used in [2] is not appropriate. In fact, the alternative described here takes the special nature of the objective function, a minimum of linear functions, into account and not only overcomes its non-differentiability but also is considerably more efficient and easier to implement than a gradient search.

We now give an outline of the optimization procedure. A high level flowchart of this procedure is given in Figure 1. As mentioned

above, at this level the procedure is almost identical to the Flow Deviation Algorithm. The key difference, which is only evident in a more detailed description, is how the optimal superposition of flows is found.

We define the length of a link to be its utilization. Initially, we set all link lengths to 0. We define the length of a path to be the length of the longest link in the path plus a small constant times the number of links in the path. This latter term is added to break ties among paths with equally utilized links in favor of a path with the smallest number of links. Note that this definition of path length is different from the conventional one but it serves our purpose. Shortest paths using this metric are computable using conventional shortest path algorithms.

The routing pattern found at each stage in the optimization procedure is a single shortest path for each requirement r_{ij}. (In general, this path is not unique, but this poses no problem.) A flow pattern is defined as the total flow in each link and is found by loading the requirements onto the links specified in the current flow patterns.

An optimal superposition of the current flow pattern with all previous flow patterns is then found. This is the key step in the procedure and is done by finding the value of λ between 0 and 1 which minimizes V, the maximum link utilization, where λ represents the fraction of all previous flow patterns used. The new optimal superposition of flows is then λ times the previous superposition plus $(1-\lambda)$ times the current flow pattern. A new superposition, and hence link utilizations, is then obtained. This in turn yields a new value for V(K) and the link lengths. We can now start another iteration. If, however, no improvement in V(K) has been observed, the iteration has converged

and we terminate the procedure with an optimal flow pattern. By saving the routing patterns and $\lambda^{(K)}$, the values of λ for each K, the optimal routing pattern can be obtained. In particular, if $P_{ij}^{(K)}$ is the (i,j) path first used in the routing pattern in iteration K then the fraction of commodity (i,j) using $P_{ij}^{(K)}$ is

$$(1-\lambda^{(K)}) \prod_{j=K+1}^{M} \lambda^{(j)}$$

where M is the number of iterations and the product is defined equal to 1 for K = M.

We now turn to the problem of how to find the optimal superposition of flows. Consider two flow patterns, $F^{(1)}$ and $F^{(2)}$. Each flow pattern assigns a flow to each link. Thus $f_{ij}^{(1)}$ and $f_{ij}^{(2)}$ are the flows assigned to link (i,j) in $F^{(1)}$ and $F^{(2)}$, respectively. For any λ between 0 and 1, the flow assigned to link (i,j) by superposing $\lambda F^{(1)}$ and $(1-\lambda)F^{(2)}$ is then

$$\lambda f_{ij}^{(1)} + (1-\lambda) f_{ij}^{(2)}$$

which equals

$$f_{ij}^{(2)} + \lambda(f_{ij}^{(1)} - f_{ij}^{(2)})$$

which is a linear function of λ. Dividing by C_l, to obtain utilizations, there will be, in general, a different function, $a_l + b_l\lambda$ for each link l. (We will for simplicity refer to links by a single index, l, rather than endpoints (i,j).)

We seek the value of λ which minimizes the maximum of these functions over all l. Several simple observations allow us to find this value of λ in an efficient and straightforward manner. First, if for two

links, l and m, $a_l \geq a_m$ and $b_l \geq b_m$ then link l is said to dominate link m and link m can be ignored as it clearly does not participate in the maximum since $a_l + b_l\lambda \geq a_m + b_m\lambda$ for all values of λ. Indeed, if $a_l + b_l\lambda \geq a_m + b_m\lambda$ for all λ between 0 and 1 then link m may be ignored. Note that this latter condition is not equivalent to the former, for example if $a_l = 10$, $b_l = 0$, $a_m = 2$, and $b_m = 3$.

We can then arrange the links l in descending order of a_l and examine the b_l. Any link m for which b_m does not exceed b_l, where l is the predecessor of m in the order, can be eliminated. We now have an ordering of links which is descending in a_l and ascending in b_l. We then compute, for each adjacent pair of links l and m, the value λ_l for which $a_l + b_l\lambda_l = a_m + b_m\lambda_l$. The λ_l should form an ascending sequence. A value of λ_l which is less than the value of its predecessor in the sequence corresponds to a link l which can be eliminated from further consideration. In this case, link l is eliminated and m has a new predecessor. The λ_p is then recomputed for link p the new predecessor of m and this process is repeated. For link n, the last link in the sequence, $\lambda_n = 1$.

We now compute for each remaining link l $v_l = a_l + b_l\lambda_l$ and select the minimum of these values. The resulting λ_l and v_l are the desired values yielding the optimal superposition.

This entire process is illustrated in Figure 2. The links have been sorted so that the a_l are descending. Links with nonascending b_l have already been eliminated. Thus the b_l form an ascending sequence. This is evident in Figure 2 by the fact that the lines form a sequence increasing in slope. The intersection of lines 1 and 2 (i.e. the lines starting at a_1 and a_2) defines λ_1. Similarly, the intersection of lines

2 and 3 defines λ_2 and $\lambda_2 > \lambda_1$. So thus far no line is dominated. The intersection of lines 3 and 4 takes place between $\lambda = 1$ so line 4 is dominated by line 3 and line 4 is thus eliminated from further consideration. The intersection of lines 3 and 5 defines a value of λ_3 (dotted line), but when λ_5 is computed we find it to be less than λ_3. So, line 5 is dominated by line 6 and removed from further consideration. λ_3 is then recomputed from the intersection of lines 3 and 6. λ_6 is computed from the intersection of lines 6 and 7. Finally $\lambda_7 = 1$. This leaves us with λ_1, λ_2, λ_3, λ_6, and λ_7 (also $\lambda_0 = 0$). We search among the corresponding v_1 and find v_3 is minimum. It and λ_3 define the desired superposition.

The entire optimization process is illustrated in Figures 3, 4, and 5. The network consisting of 3 nodes, 6 links and 6 requirements is shown in Figure 3. For simplicity, we assume symmetric requirements and link capacities. We can thus assume a symmetric solution, i.e., routes for r_{ij} the reverse of routes for r_{ji} and equal utilization of each link in both directions. This allows us to only consider 3 links and 3 requirements in the example. This is done to simplify the example. The actual procedure works with directed links and requirements. It can also be used with undirected links and requirements but such a situation is rarely physically meaningful.

Initially, all requirements are routed directly since the initial shortest paths by our definition would be the paths with the minimum number of links. This is illustrated in Figure 4a. The link lengths are then recomputed -- link 1 has a utilization of .3 and hence a length of .3, etc. The shortest paths are then recomputed and are shown in Figure 4b. Note that the shortest path from B to C is now B-A-C.

The requirements are loaded onto these paths. The flow pattern is shown in Figure 4c.

Now a superposition of the flow patterns in Figures 4a and 4c is done. Figure 5 illustrates the dynamics of this. Note that r_{AC} dominates r_{AB} and that the optimal λ is .9 and $v = 4.5$. Figure 4d shows the resultant routing pattern formed by using the first routes for 90% of the traffic and the second routes for the remaining 10%. Figure 4e shows the flow pattern resulting from the superposition. Note that the maximum utilization is .45 (in links (A,C) and (B,C)) which is less than the maximum in either of the patterns in Figures 4a and 4c.

We now recompute the link lengths and the shortest paths. The resultant routes are the same as in Figure 4a. An optimal superposition between this flow pattern and the one in Figure 4e is then done. The optimal value of λ is 1, no improvement in v is found and we conclude that the routing and flow pattern in Figures 4d and 4e are optimal. Note that the links (A,C) and (B,C) are both maximally utilized. They form a cut which is analogous to the saturated cut in Gerla's Cut Saturation Method. (The existence of such a cut is a necessary condition for the optimality of a flow pattern.)

We thus have developed a simple and efficient algorithm for obtaining stable flow patterns for use globally as the higher level in our 2 level adaptive routing procedure. In the coming months we hope to implement this procedure and experiment with it.

In an allied study we investigated a pattern for placing virtual calls on a network. A simulation program was written to directly observe the dynamic performance of an algorithm which loads calls on alternate routes according to the following algorithm:

1. Load each incoming call onto the route currently carrying the smallest number of calls. (The number of calls carried by a route is defined for the purposes of this algorithm to be the number of calls on the first link in the route.)
2. If there is a tie among several routes in a set, S, select route i with probability $P_i(S)$.

The simulation was written to provide us with a first glimpse of the dynamic performance of such a procedure as a guide for further research in this area. We thus wanted to keep it as simple as possible and considered a 3 node 6 link network as shown in Figure 5a with symmetric requirements. The program can easily be expanded to consider more general cases but we chose this simple one initially in order not to obscure the basic results.

Calls arrive at each node at a rate λ (Poisson) and are served at rate μ (exponential) by the links, i.e., have exponential duration with average length $1/\mu$. Each call has a choice of a 1 hop path or a 2 hop path. A call arriving at a node is equally likely to be destined for either other node. Thus, there is total symmetry in the system. It should be noted that a call taking a 2 hop path occupies 2 links but remains in the system for time $1/\mu$ on average (not $2/\mu$).

The simulation is straightforward. Call arrivals are generated randomly and arriving calls are routed according to the algorithm given above. The number of calls taking the 1-hop and 2-hop routes were recorded for each run. A parameter, α, determined the probability of taking the 1-hop route when there was a tie between the 2 routes (α = Prob {using the 1-hop route in case of a tie}).

For $\alpha = .5$ the fraction of calls taking the 1-hop route was, not surprisingly, very close to ½. For $\alpha = 0$, however, the fraction varied. For $\lambda/\mu = 1$, 80% of the calls took the 2-hop path. For $\lambda/\mu = 10$, 66% of the calls took the 2-hop path. For $\lambda/\mu = 50$, 65% of the calls took the 2-hop path. For $\alpha = 1$, the results (fraction on 1-hop versus fraction on 2-hop paths) reversed relative to the results for $\alpha = 0$.

We thus conclude that we have some control but not total control over the routing via α which only operates during a tie. The control gets greater for systems with a smaller number of calls in progress, as evidenced by the results for smaller values of λ/μ (which is directly related to the number of calls in the system). Observations of the number of calls in the system at various points in a simulation run led to the conclusion that the system is stable, i.e., that the link loads reach a stable level and remain close to that point and close to one another.

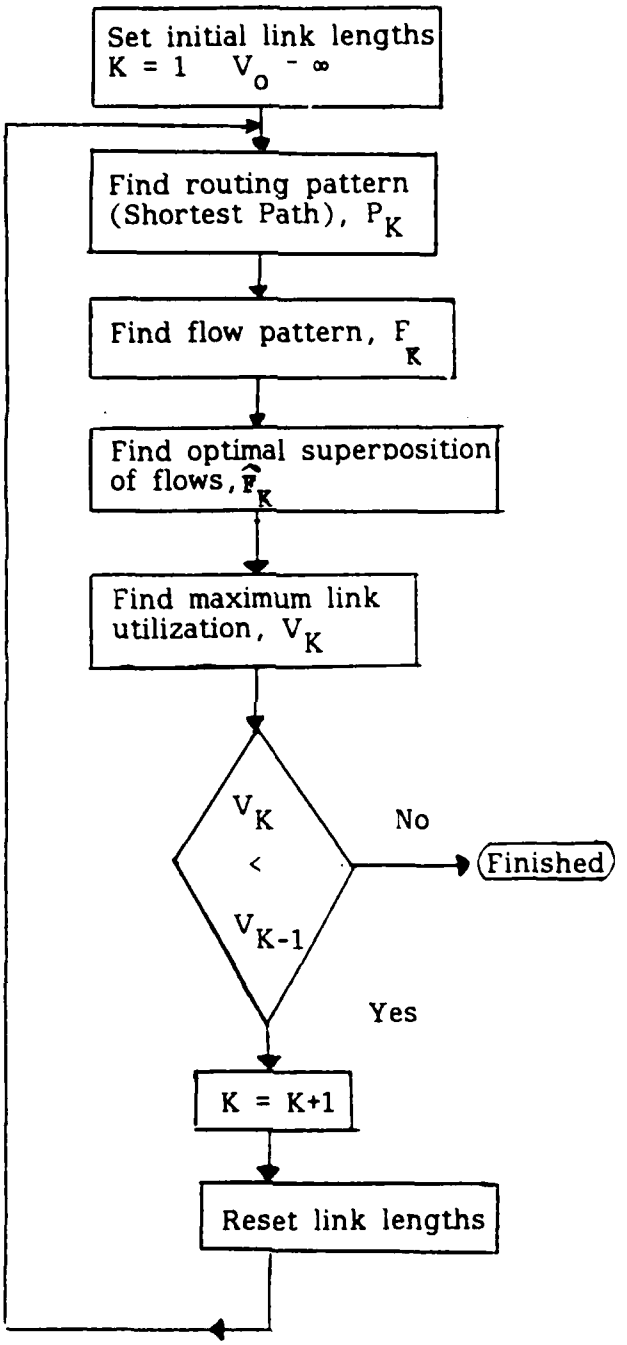


FIGURE 1

OPTIMIZATION PROCEDURE

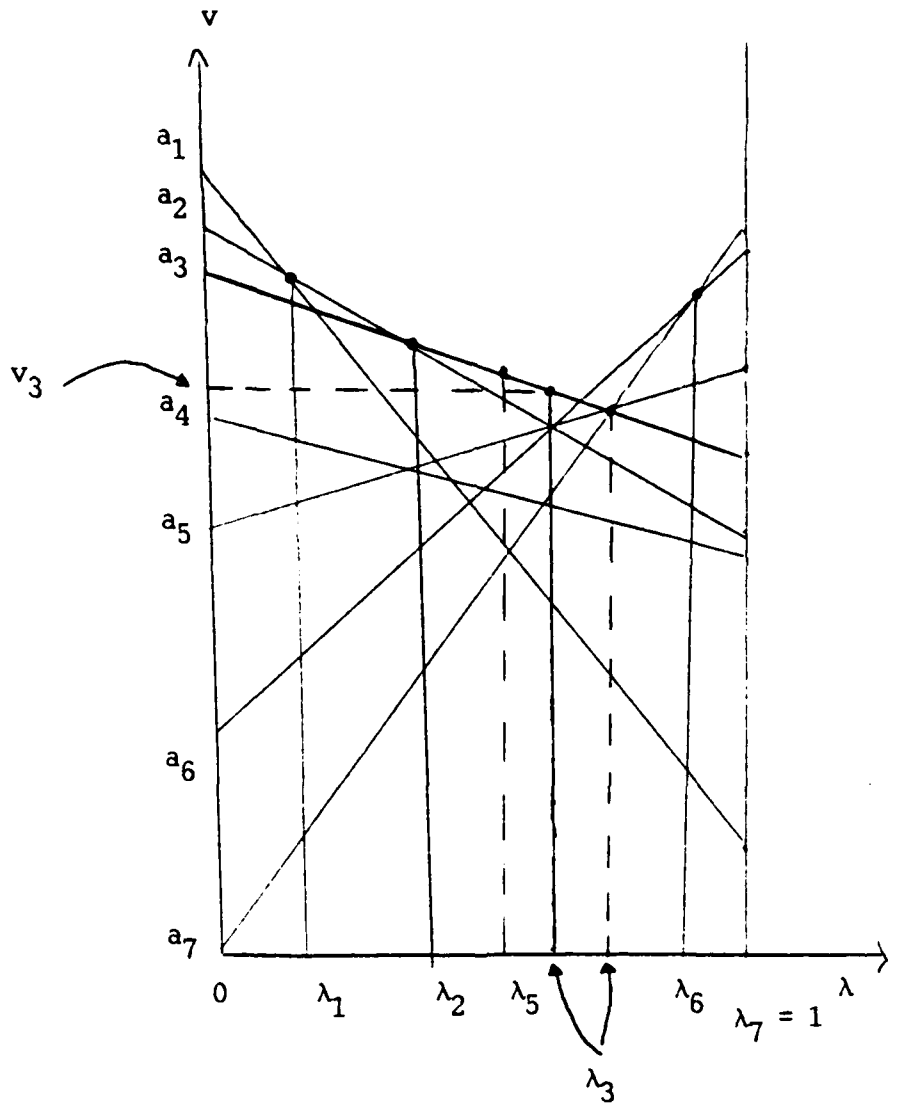
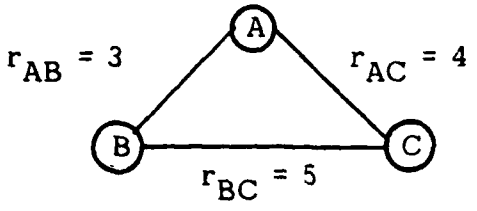


FIGURE 2

FLOW SUPERPOSITIONS



$C_{AB} = C_{AC} = C_{BC} = 10$

FIGURE 3

NETWORK AND REQUIREMENTS

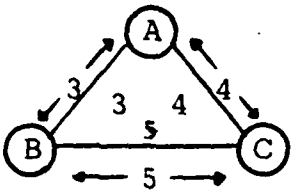


FIGURE 4a
Routing Pattern 1
= Flow Pattern 2

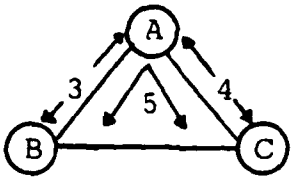


FIGURE 4b
Routing Pattern 2

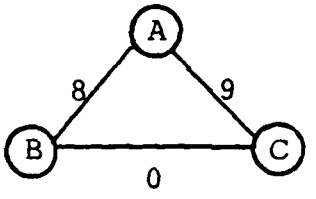


FIGURE 4c
Flow Pattern 2

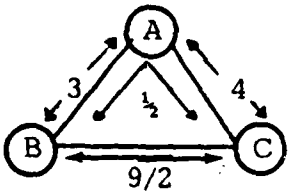


FIGURE 4d
Superposition
of Routing
Patterns

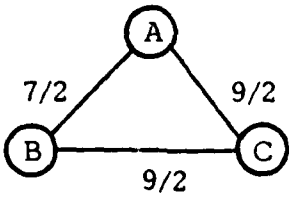


FIGURE 4e
Superposition
of Flows

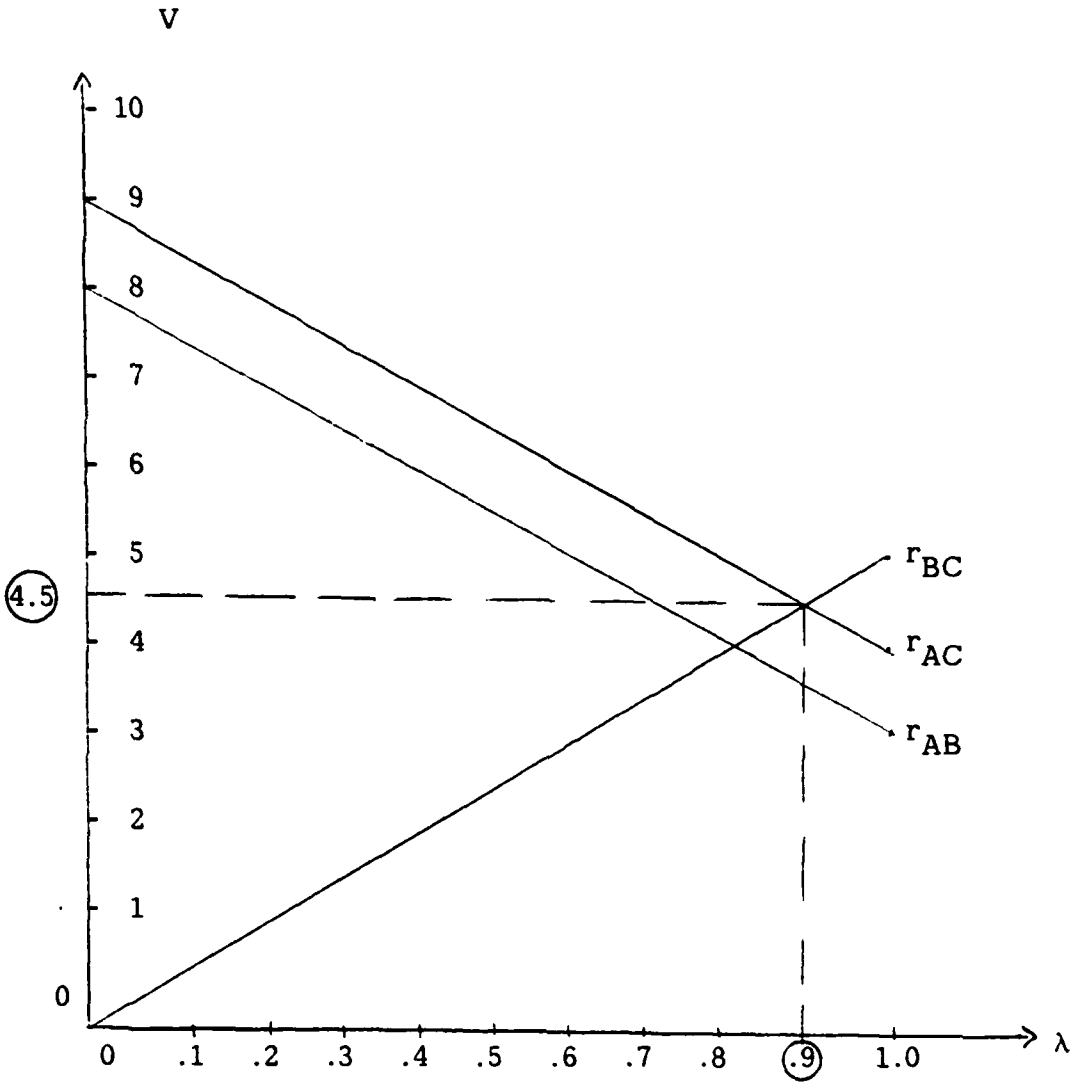


FIGURE 5

MAXIMUM LINK FLOW (V) VERSUS λ

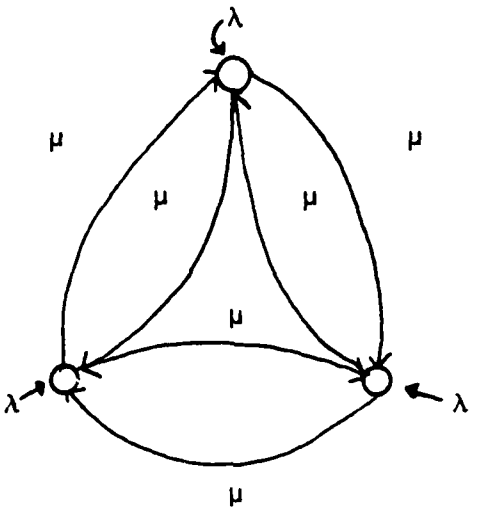


FIGURE 5a

A SYMMETRICAL NETWORK

A.2 Locally Adaptive Routing

For routing at the local level we proposed a priority routing algorithm [1]. Its function is summarized in Figure 6 whereby dedicated traffic, i.e. those without choice, join queues Q_1 and Q_2 in order to obtain service on links A and B respectively. Traffic with choice join the middle queue Q_3 and is served on either A or B whenever Q_1 or Q_2 is empty. Assuming arrival rates $\lambda_1, \lambda_2, \lambda_3$ and equal service times μ, the service utilizations ρ_A and ρ_B are given in the Figure. The parameter α, denotes the fraction of the non-dedicated traffic which is served by A. A heuristic estimate for α is given by

$$\alpha \cong \frac{1 - \rho_A}{2(1 - \rho)}$$

Solving for α we obtain

$$\alpha \cong \frac{1 - \rho_1}{2 - \rho_1 - \rho_2} \quad .$$

This assumes that packets from λ_3 join A in proportion to the probability that Q_1 is empty. It has been shown that, by serving the non-dedicated traffic in idle periods of Q_1 and Q_2, the system achieves close to two-server (M/M/2) behavior, thus reducing the packet delay (averaged over the three queues) by a factor approaching 2 when compared to two independent M/M/1 queues. The latter models the random bifurcation with the non-dedicated traffic joining the ends of queues Q_1 and Q_2 according to a random rule. The introduction of the middle queue is the essential key to improving delays.

An additional form of control exists in our priority rule, and may be used to achieve certain values of ρ_A and ρ_B without destroying the

multiserver behavior. This is possible, by taking advantage of non-dedicated arrivals to a totally empty system. Then both servers are idle, and the packet can be sent to A or B with preset probabilities β and $1 - \beta$, respectively.

Let P_E = Prob. {Both servers are idle}

Then Pr {only A is idle} = $1 - \rho_A - P_E$

Pr {only B is idle} = $1 - \rho_B - P_E$

Pr {at least one server is idle} = $2(1 - \rho) - P_E$

The fraction of non-dedicated traffic which joins A can be modified to

$$\alpha \cong \frac{1 - \rho_A - P_E + \beta P_E}{2(1 - \rho) - P_E} = \frac{1 - \rho_1 - P_E(1 - \beta)}{2 - \rho_1 - \rho_2 - P_E}$$

(Our previous results, apparently ignoring β, actually assumed $\beta = 1 - \rho_2/2-\rho_1-\rho_2$.)

Since P_E cannot be evaluated in a closed form, we obtain bounds. An upper bound is the M/M/2 probability of an empty system. This is true since such a service mechanism assumes that all λ_1, λ_2, λ_3 traffic can use either server and thus achieves the best utilization. A lower bound is for an M/M/1 queue. Thus

$$(1 - \rho_A)(1 - \rho_B) \leq P_E \leq \frac{1 - \rho}{1 + \rho} .$$

A looser but more usable lower bound is $1 - 2\rho$.

As an example let $\rho = \frac{1}{4}$. Then $\frac{1}{2} \leq P_E \leq 3/5$. If $\rho_1 = \rho_2 = \rho/2$, then for $\beta = 1$, $\alpha \geq 7/10$ and for $\beta = 0$, $\alpha \leq 3/10$. The range of control using β is at least 40% of the total.

(Numerical results to come)

The interaction between the global quasi-static routing and the locally adaptive strategy lies in determining average link flows at the high level and designing the local policy in order to achieve them. In terms of the node model of Figure 7, the local problem can be formulated as follows:

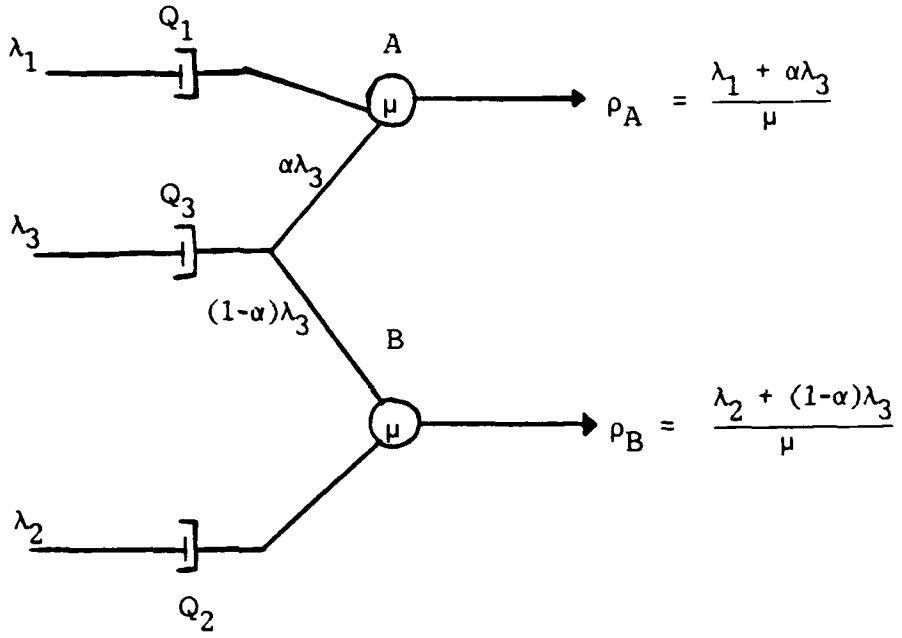
Given the input traffic λ_1, λ_2 (dedicated) and λ_3 (non-dedicated) and the desirable output traffic λ_A, λ_B, where $\lambda_1 + \lambda_2 + \lambda_3 = \lambda_A + \lambda_B$, minimize the average queuing delay. In the preceding section, we demonstrated how we can control λ_A to a certain degree using our priority scheme and adjusting β (the probability of routing a packet for A when both servers are idle). The impact of β can be significant especially under low utilizations, but cannot always achieve the whole spectrum of λ_A between λ_1 and $\lambda_1 + \lambda_3$. As shown in Figure 7, at the extreme values, the non-dedicated traffic becomes fully dedicated and delays are that of two independent M/M/1 queues. At values of λ_A in the middle of the allowable range, variations of β will provide the desirable values with delays close to the M/M/2 lower bound. At the extremes, we may have to violate the priority scheme and either defer service of a non-dedicated packet in order to control its flow, or occasionally break the priority rule in order to serve a non-dedicated packet. Our objective is still to be able to fill-in idle periods of Q_1, Q_2 with non-dedicated packets as much as possible in order to minimize delays. From a preliminary investigation of the problem, we arrived at the following observations:

A. No action need be taken while the system remains in a state. Actions may have to be taken at transition times of the state vector. By actions we mean deferment of service or breaking

of priority. By transition times we mean any arrival or departure time in any of the queues and servers of the system. This observation follows from the Markov nature of the system.

B. A countable number of points in the optimal delay vs. λ_A curve can be obtained via deterministic policies, whereby actions are assigned to states with probability one. This follows from formulating the truncated state space problem as a finite state Markov Decision process with average cost minimization under a state frequency constraint (the specified value of λ_A). It is known that such problems lead to probabilistic (randomized) action-state assignments in general [3]. However by incorporating the constraint within the objective function (average delay) using a Lagrange multiplier, the problem becomes an unconstrained optimization problem, which leads to deterministic policies [4]. Due to the discrete finite state and action spaces, only a finite number of optimal points can be found using Lagrange multipliers.

A simplified version of the node optimization problem can be obtained by deleting the dedicated queues Q_1, Q_2. Then the problem can be formulated as an optimal control of output flows in an M/M/2 queue as in Figure 8. The same arguments mentioned above indicate the nature of the optimal policy as shown in the figure. We do have strong indications that the deterministic policies are simple threshold schemes, whereby packets defer using server B (if $\lambda_A > \lambda_B$) unless the number of packets waiting in queue Q exceeds a threshold K. Values of λ_A in-between two threshold policies can be obtained by implementing a random choice at the threshold.



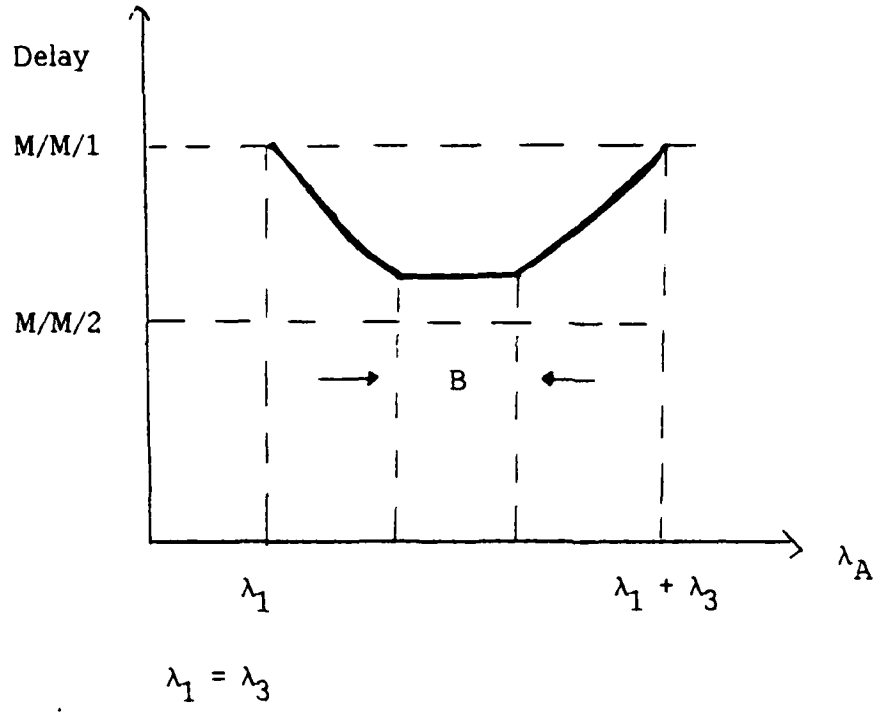
$$\alpha \simeq \frac{1 - \rho_A}{2(1-\rho)} \qquad \rho = \frac{\lambda_1 + \lambda_2 + \lambda_3}{2\mu}$$

$$\rho_1 = \lambda_1/\mu$$

$$\rho_2 = \lambda_2/\mu$$

FIGURE 6

FLOWS IN PRIORITY ROUTING



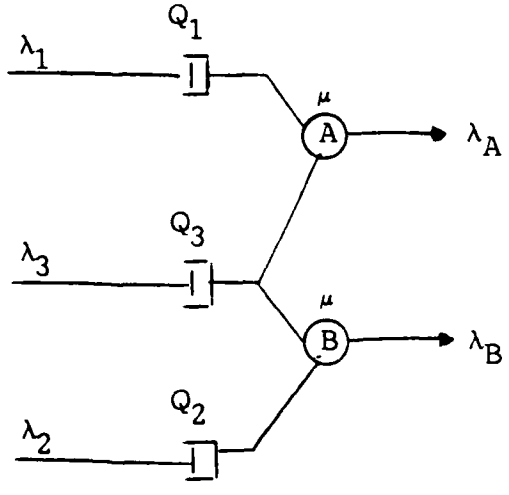
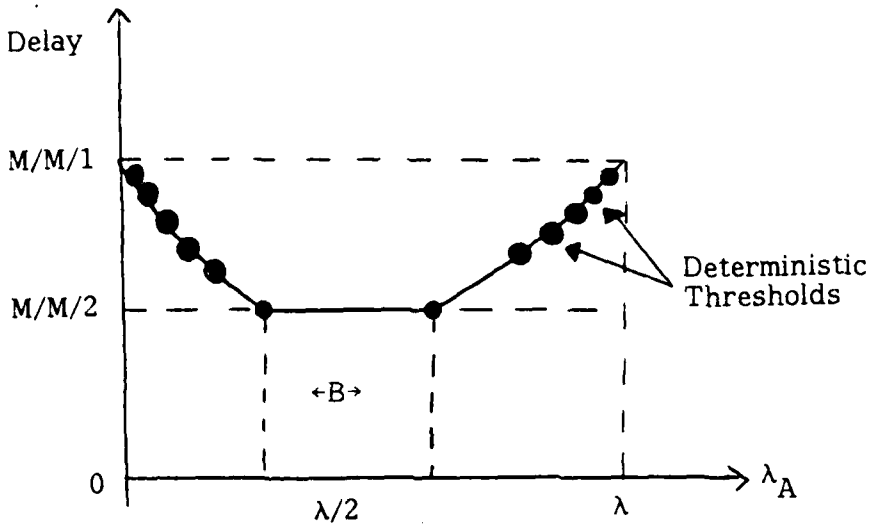


FIGURE 7

OPTIMUM DELAY WITH GIVEN λ_A



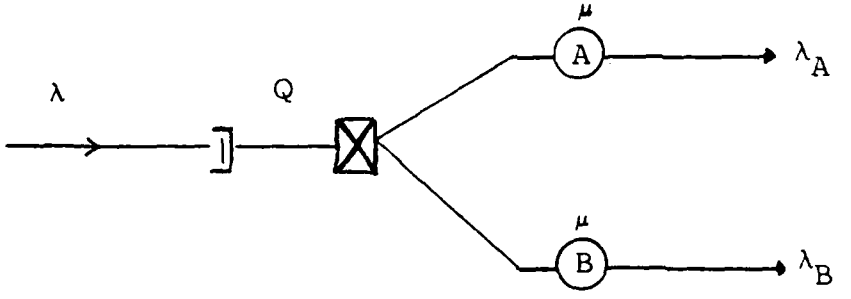


FIGURE 8
OPTIMUM DELAY IN M/M/2 QUEUES WITH SPECIFIED λ_A

B. Multiple Access Techniques with Arbitrary Packet Length Distributions

B.1 Introduction

In our original multihop packet radio analysis, we assumed exponentially distributed packet lengths. We have been able to generalize the analysis for packet lengths having densities formed by the positive sum of exponential terms (see Appendix C). In our analysis we assumed that propagation delays among neighboring PRU's are negligible. Thus in a Carrier Serving Multiple Access (CSMA) mode of operation, collisions may occur due to the "hidden terminal" phenomenon only (i.e., two non-communicating PRU's schedule packet transmissions to a common neighbor simultaneously).

CSMA analyses incorporating the effects of propagation delays have been reported extensively in the literature for single-hop networks (i.e., all PRU's hear each other) and fixed packet sites. As a first step in generalizing these results, we studied single-hop multiple access protocols with non-fixed packet lengths. Although our main thrust is on CSMA packet radio, we also derived formulas for pure ALOHA and CSMA with collision detection (CSMA/CD). The former was a necessary step in order to demonstrate the impact of packet length distribution on the simplest multiple access method, whereas the latter is a straightforward extension of pure CSMA and is especially popular in local networking environments. Note that the pure ALOHA case was studied previously [5], whereas no extension has been reported on CSMA to our knowledge. The CSMA/CD result is so simple that it may already be known.

In what follows, we summarize the variable packet length analyses in pure ALOHA, CSMA and CSMA/CD. In all cases we assumed infinitely many Poisson sources and Poisson aggregate scheduling processes, with rates s and g packets/sec respectively. Packet lengths are distributed arbitrarily.

B.2 Pure ALOHA

Referring to Figure 9, we consider a transmission of length Y (shaded). This transmission will be successful if a) no other packet is transmitted in Y seconds and b) no previously transmitted packet is still transmitting. We are assuming zero capture. Calling these probabilities P_a and P_b, we obtain $s = gP_aP_b$. But

$$P_a = \int_0^\infty e^{-gy} \, fY(y) \, dy = MY(-g)$$

where

$$MY(g) = \int_0^\infty e^{gy} \, fY(y)dy$$

is the moment generating function of Y and fY(y) is its density. P_b is found by considering the T second interval prior to the transmission in question. Assume transmissions in that interval occur T_i seconds before the start of our test transmission and have length Y_i. Then

$$P_b = \lim_{T\to\infty} P(\text{all } T_i \geq Y_i)$$

But the number of transmissions in T is Poisson and all are identically distributed and independent. Therefore

$$P(\text{all } T_i \geq Y_i) = \sum_{k=0}^{\infty} [P(T_i \geq Y_i)]^k \frac{(gT)^k}{k!} e^{-gT}$$

$$= e^{-gT[1-P(T_i \geq Y_i)]} = e^{-gTP(T_i < Y_i)}$$

Here T_i is uniform in the interval $(0,T)$ and Y_i is distributed as Y. They are independent. Thus

$$P(T_i < Y_i) = \frac{1}{T} \int_0^T [1 - F_Y(t)]\, dt$$

and

$$TP(T_i < Y_i) \to \int_0^{\infty} [1 - F_Y(t)]\, dt = E(Y)$$

Here $F_Y(y)$ is the distribution function of Y and E(Y) its expectation. Finally we have

$$s = gM_Y(-g)\, e^{-gE(Y)} \quad .$$

Note that if Y is fixed then $s = ge^{-2gy}$ as it should.

But Y is the length of the transmitted packets. Condition (b) above does not involve the length of the transmitted packet Y. But condition (b) does! Longer packets are more likely to suffer a collision. Let X be the length of the offered (or successful) packets. Y should in a sense be larger since longer packets are retransmitted more often. Due to condition (a) alone a packet of length x will be successfully transmitted with probability e^{-gx} and requires an average of e^{gx} transmissions to be successful. Thus

$$f_Y(y) = e^{gy}\, fx(y)/M_X(g).$$

Also

$$E(Y) = \frac{dM_X(g)}{dg} / M_X(g)$$

and $P_a = 1/M_X(g)$.

Thus $$s = \frac{g}{M_X(g)} e^{-gM_X'(g)/M_X(g)}$$

This also reduces to $s = ge^{-2gx}$ when $M_X(g) = e^{gx}$. This result has been already established [5] but is derived here in a different manner.

B.3 CSMA

Refer to Figure 10 for CSMA. After an idle period a packet scheduled with rate g is transmitted. The packet lasts for X seconds. In the propagation time a after transmission any other scheduled packet can also be transmitted thus causing a collision. We assume $X \geq a$. Again we assume zero capture. If no such packet is transmitted, then the original transmission is successful, lasts for X seconds, and is followed by an a second period to clear the channel and the idle state resumes. Thus the successful rate is

$$s = ge^{-ga} \text{ P(channel is idle).}$$

But $$\text{P(channel is idle)} = \frac{1/g}{1/g + a + E(Z)}$$

where 1/g is the average idle time and Z is the busy period exclusive of the last a seconds.

To evaluate E(Z) we denote the times of the transmissions of interfering packets as T_i and their lengths as X_i. The number of such packets is exponential. Collisions depend only upon the propagation time a and not on the length of the transmitted packet as in ALOHA. Thus

$$Z = \max \{X, T_i + X_i\}.$$

and $$F_Z(z) = F_X(z) \sum_{k=0}^{\infty} [F_W(z)]^k \frac{(ga)^k}{k!} e^{-ga}$$

where $W = T_i + X_i$, T_i is uniform in (0,a) and is independent of X_i, which is distributed as X. Thus

$$F_Z(z) = F_X(z)\, e^{-ga[1 - F_W(z)]}$$

and $$E(Z) = \int_0^{\infty} [1 - F_Z(z)]\, dz.$$

The last two equations can be used to find E(Z), although not easily. Finally

$$s = \frac{ge^{-ga}}{1 + ga + gE(Z)} .$$

B.4 CSMA/CD

Refer to Figure 11 for CSMA/CD (unslotted). Here collisions are detected and transmissions aborted. Again packets are scheduled with a rate g. After an idle period a packet is transmitted. If no packets are transmitted in the next a seconds that packet is successful. After a seconds to clear, the channel returns to idle. A scheduled packet can be transmitted in the first a seconds and will cause a collision. But it will be aborted a seconds after the original transmission. The original transmission will be aborted a seconds after the start of the colliding packet. Thus, as before,

$$s = ge^{-ga} \text{ P(channel idle)}.$$

$$\text{P(channel idle)} = \frac{1/g}{1/g + a + E(Z)}$$

and $Z =$ X, with prob. e^{-ga}

$\min\{T_i\} + a$, with prob. $1 - e^{-ga}$.

Here we assume, for convenience, that $X \geq 2a$. Solving we get

$$E(Z) = E(X)e^{-ga} + \sum_{k=1}^{\infty} \left(\frac{a}{k+1} + a\right) \frac{(ga)^k}{k!} e^{-ga}$$

$$= E(X)e^{-ga} + \frac{1}{g}\left[1 - (1+ga)e^{-ga}\right] + a(1 - e^{-ga})$$

or

$$s = \frac{ge^{-ag}}{1 + gE(X)e^{-ga} + (1+2ga)(1-e^{-ga})}$$

Note than only E(X) appears in the above equation.

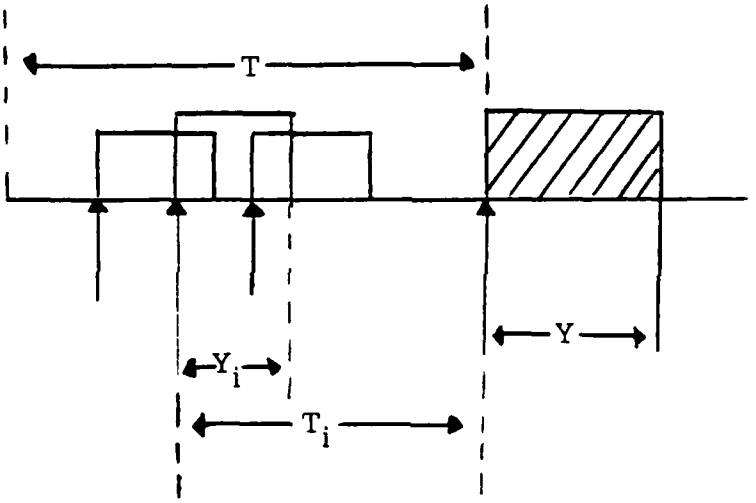


FIGURE 9

PURE ALOHA

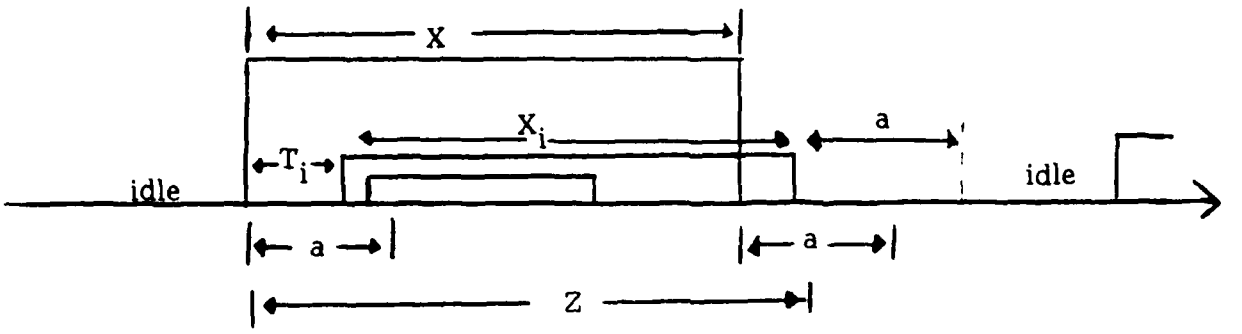


FIGURE 10

CSMA

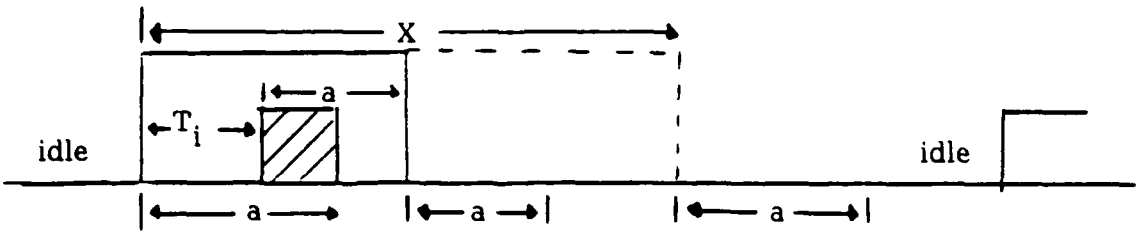


FIGURE 11

CSMA/CD

REFERENCES

1. R.R. Boorstyn and A. Livne, "A Technique for Adaptive Routing in Networks," IEEE Transactions on Communications, April 1981.

2. D.G. Cantor and M. Gerla, "Optimal Routing in a Packet-Switched Computer Network," IEEE Transactions on Computers, October 1974.

3. C. Derman, Finite State Markovian Decision Processes, Academic Press, 1970.

4. S.M. Ross, Applied Probability Models with Optimization Applications, Holden-Day, 1970.

5. S. Bellini and F. Borgonovo, "On the Throughput of an ALOHA channel with Variable Packet Lengths," IEEE Transactions on Communications, November 1980.

PERSONNEL

Robert R. Boorstyn, Professor
Aaron Kershenbaum, Associate Professor
Basil Maglaris, Assistant Professor

Students

William Chuang (Algorithms)
Veli Sahin (Multihop Packet Radio)
David Tsao (New Switching Techniques)

Rachel Mendelsohn (Algorithms)
William Chen (Multihop Packet Radio)
H.K. Chao (Message Delay in Networks)

PAPERS PUBLISHED AND WRITTEN

A. Kershenbaum, "Generalized Augmenting Paths for the Solution of Combinatorial Optimization Problems," IFIPS

J.F. Hayes & R. Boorstyn, "Delay and Overhead in the Encoding of Data Sources," IEEE Trans. Communications, pp. 1678-1683, Nov. 1981

P. Chu, R. Boorstyn, and A. Kershenbaum, "A Simulation Study of a Dynamic Routing Scheme," NTC '81, pp. A3.4.1 to 11, November 1981.

B. Maglaris, T. Lissack, and M. Austin, "End-to-End Delay Analysis on Local Area Networks: An Office Building Scenario," NTC'81, pp. A2.3.1 to 6, Nov. 1981.

A. Kershenbaum, "A Note on Finding Shortest Path Trees," Networks Journal, vol. 11, pp. 399-400, 1981

A. Kershenbaum and R. Boorstyn, "Centralized Teleprocessing Network Design," to appear in Networks Journal.

ACTIVITIES

The following Ph.D. students completed their studies.

William Chuang, Probabilistic Analysis of Algorithms
Veli Sahin, Analysis of Multihop Packet Radio Networks
David Tsao, Comparison of Switching Techniques

Prof. Basil Maglaris joined our faculty and our research team.

Prof. Kershenbaum became Associate Editor for the Journal of Telecommunication Networks.

Prof. Boorstyn is on the Standing Committee for the INFOCOM conferences.

Prof. Boorstyn talked on Analysis of Multihop Packet Radio Networks at Bell Laboratories and on Adaptive Routing at CCNY.

APPENDIX C

*THROUGHPUT ANALYSIS OF MULTIHOP PACKET RADIO NETWORKS**

Robert R. Boorstyn and
Aaron Kershenbaum

Polytechnic Institute of
New York
333 Jay Street,
Brooklyn, New York 11201

Veli Sahin

Bell Laboratories
Holmdel, New Jersey 07733

ABSTRACT

We consider the problem of obtaining exact expressions for throughput and blocking probabilities in multihop packet radio networks operating under CSMA. We obtain exact results for a general class of message lengths, for general topologies, and for perfect capture. These results are obtained by assuming perfect acknowledgments.

I. INTRODUCTION

We consider the problem of obtaining exact expressions for throughput and blocking probabilities in multihop packet radio networks operating under carrier sense multiple access (CSMA). Procedures are developed which can be used to analyze general topologies for a general class of packet length distribution. Examples of chains, rings, and stars are presented.

II. THE NETWORK MODEL

We consider the problem of analyzing the throughput capability of a multihop packet radio network operating under carrier sense multiple access (CSMA). Thus, we assume that the network is comprised of terminals equipped with radio transponders suitable for broadcasting data over a limited distance. In general, the source and destination terminals

* This research was partially supported by USARMY CENCOMS under contract DAAK 80-80-K-0579, and by the National Science Foundation under grant ENG-79-08210.

cannot hear each other directly, and the data has to be relayed by one or more intermediate devices. A separate set of devices, called repeaters, may exist for this purpose, or the terminals themselves may relay messages for one another.

Control of the network is completely distributed, i.e., no station or central control mechanism is assumed to exist. Rather, we assume that each source terminal has prestored one or more routes to all destinations and includes all necessary routing information in the packets if transmits. These assumptions are made merely to simplify the presentation. In fact, the results presented are valid for networks using alternate routing as long as routing changes are not made over short time intervals. One of the motivations for this study came from a consideration of the design of routing procedures for such networks. It was necessary, however, to first develop an understanding for the throughput of various topologies.

Exogenous traffic is modeled as independent Poisson processes arriving at each source node, with appropriate rates and packet lengths. The topology is specified by a listing of which terminals (or repeaters) can hear each other. In the remainder we will not distinguish between terminals and repeaters and will refer to them collectively as either terminals or nodes. In general the transmissions of one terminal can be heard by many other terminals. The routing will specify which terminal is to repeat the packet, if necessary.

If two or more transmissions are simultaneously heard by a terminal (called a "collision") at least one, and possibly both, is "lost" and must be retransmitted. We assume retransmissions are scheduled at a random instant in time sufficiently far in the future so as to preserve the Poisson nature of the combined traffic stream, which now consists of exogenous traffic and rescheduled traffic. For this study, we assume that a packet can be retransmitted as many times as is necessary, i.e., that there is no maximum allowable number of retransmissions.

At any time, terminals may either transmit or receive, they cannot do both simultaneously. Before transmitting, a terminal senses the channel. If it detects that any of its

neighbors (i.e., terminals that it can hear) are transmitting (by, e.g., sensing a carrier) it reschedules the transmission as for collided packets above. If at the scheduled time for a transmission, the terminal is already engaged in transmitting a packet, the new packet is also rescheduled as above. Thus packets are continually rescheduled until they are successfully delivered to the next terminal on their route. We assume that the total stream of traffic scheduled by any terminal is a Poisson process. This includes originating traffic and packets rescheduled either due to collisions or due to the channel having been sensed busy. This scheme is called carrier sense multiple access (CSMA). The Poisson assumption is valid for the assumptions made above and will yield accurate results for throughput. Compromises will have to be made, however, if an accurate picture of time delays is to be considered.

It is possible, due to non-zero propagation delay, that collisions of transmissions from neighboring terminals may still take place despite the CSMA strategy. This will occur if a terminal senses the channel before another terminal's transmission is received. This effect is small if terminals are reasonably close or are not transmitting at high speed. We will ignore this phenomena here, and assume that all transmissions are instantly heard by their neighbors.

A passive acknowledgment is used for transmission to neighbors. The transmitting terminal listens to the channel to hear if a packet is being rebroadcast by a neighbor. If after a prespecified time interval, the transmitting node does not hear the packet rebroadcast, it retransmits the packet. But the packet may have been successfully received by the neighbor even though the originator does not hear the rebroadcast. Duplicate packets may be transmitted and deleted only at the final destination or they may be detected and deleted earlier. An end-to-end acknowledgment is returned to the originator from the final destination. In this paper we assume that passive acknowledgments are always heard and ignore the effect of end-to-end acknowledgments. Alternately, these acknowledgements could have been added to the required traffic. (In a sequel paper, the effect of passive acknowledgements will be studied).

We depict the topology of the network by a graph where terminals are represented

by nodes. The nodes are connected by a link if they can hear each other's transmissions, i.e., if they are neighbors. As an example see Figure 1. Node A can hear node B, but not node C. Node B can hear both nodes A and C. Node C can hear node B, but not node A. If node A is transmitting to node B and node C begins transmitting, then the transmission from A to B may be lost depending upon the "capture" assumptions we make. A conservative assumption is that the A to B transmission is lost - this is known as zero capture. Alternatively, perfect capture assumes that this transmission is successfully received. Half-amplitude capture assumes that the transmission is lost if C dominates A at B. This can happen if C is closer to B than A is to B, or has a greater signal strength perceived at B than A has. If A dominates C, then the transmission is successful. However, in all cases of a collision we assume the later transmission is lost. Thus if C is transmitting to B, this packet is lost in all cases. We will consider only perfect capture situations below. Note that under CSMA if node B is transmitting, neither A or C is allowed to transmit.

We assume that a routing has been specified. This takes the form of deciding which of the neighbors are to rebroadcast a packet from a particular source to a particular destination. Thus the amount of traffic that a terminal wishes to send to its neighbor can be computed. If these rebroadcast packets are scheduled at a random time far in the future the Poisson assumption for traffic streams is preserved. We assume that the traffic between neighbors is specified and form independent Poisson processes. We assume that the packet length is reassigned independently at each hop. This is analogous to the "independence assumption" in queuing networks.

The details of CSMA for a single hop network can be found in the papers by Tobagi and Kleinrock[1]. Tobagi has also developed some simple models for two-hop networks[2]. Details of a packet radio network can be found in a paper by Kahn[3]. A discussion of routing in multihop packet radio can be found in the paper by Gitman, Van Slyke, and Frank[4]. An earlier version of this paper was presented at ICC'80(5). More details can be found in the thesis of Sahin(6).

III. GENERAL RESULTS

In this section we develop some expressions that are valid for the packet radio network we have modeled above using CSMA and with an arbitrary packet length distribution. Let i be a node, n_i one of its neighbors, N_i^* the set of all the neighbors of i, and N_i the set of all i's neighbors, including i. Let g_i be the total rate (in packets/sec) of all scheduled traffic at node i. This includes originating traffic and all rescheduled traffic and is assumed to be Poisson. Let $1/\mu_i$ be the average length of packets transmitted by node i. Let $G_i = g_i/\mu_i$ be a normalized rate.

Node i is either busy (transmitting) or idle. It will transmit a scheduled packet if at the instant it is scheduled all nodes in N_i are idle. Let A be a set of nodes. Let P(A) be the probability that at a random instant all nodes in A are idle. The nodes not in A may or may not be idle. Similarly P(i), P(i,A), P(A,B) are the probabilities that i is idle, node i and nodes in A are idle, and all nodes in A and B are idle.

Since traffic is scheduled at node i with a Poisson rate g_i, will be transmitted only if N_i is idle, and transmissions have average length $1/\mu_i$, the probability that i is busy is given by

$$1-P(i) = G_i P(N_i) \tag{1}$$

If i is busy then under CSMA, n_i must be idle. Then since

$$P(n_i) = P(n_i,i) + P(n_i|i \text{ busy})[1-P(i)]$$

and $P(n_i|i \text{ busy}) = 1$, we have

$$P(n_i,i) = P(n_i) + P(i) - 1 \tag{2}$$

Similarly, if $A \subset N_i^*$,

$$P(A,i) = P(A) + P(i) - 1 \tag{3}$$

letting $A = N_i^*$ in equation (3), and using

$$P(N_i^*) = P(N_i)/P(i|N_i^*)$$

we get

$$P(i|N_i^*) = \frac{1}{1+G_i} \tag{4}$$

Equation (4) is often found in CSMA literature.

A packet from i to n_j will be transmitted when it is scheduled if N_i is idle. During the transmission all nodes in N_i^* will be idle. It will be successfully received at n_j if all neighbors of n_j not in N_i are also idle at the beginning of transmission. Otherwise a collision will occur. Let s_{i,n_j} be the rate in packets/sec, determined by the routing and assumed Poisson, of the traffic that i wishes to send to n_j. This is the required throughput or offered traffic. Let g_{i,n_j} be the rate of all scheduled traffic from i to n_j. We have also assumed that all these streams are Poisson and independent. Of these g_{i,n_j} packets per second, s_{i,n_j} must be successful. Thus

$$\frac{s_{i,n_j}}{g_{i,n_j}} = \frac{S_{i,n_j}}{G_{i,n_j}} = P(N_i, V_{n_j}) \tag{5}$$

where $S_{i,n_j} = s_{i,n_j}/\mu_i$ and $G_{i,n_j} = g_{i,n_j}/\mu_i$.

The total scheduled traffic (normalized) at a node is given by

$$G_i = \sum_{n_j \in N_i^*} G_{i,n_j} \tag{6}$$

From equations (1) through (6) we wish to derive a relation between the S_{i,n_j} and G_i, and

determine the maximum S_{l,μ_l} the network can support. This we call the (maximum) throughput or capacity. In the next section we develop this relationship for exponential packet lengths and arbitrary topologies. Later we extend the analysis to a general class of packet length distributions.

IV. EXPONENTIALLY DISTRIBUTED PACKET LENGTHS

If the packet lengths are exponentially distributed, then the system can be viewed as a Markov process where the states are identified by which nodes are idle and which are busy. Let D be a set of busy nodes. Because of CSMA, no nodes in D may be neighbors of each other. Let Q(D) be the probability that at an instant of time, all nodes in D are busy, and all nodes not in D, are idle. Then each set, D, represents a state in a Markov system, and Q(D) is the state probability. In particular, the null set $D = \phi$, represents the state that all nodes are idle.

Assume the system is in state D. It will leave the state if any $i \epsilon D$ stops transmitting. This happens with rate μ_i. Thus the transition to state {D-i} occurs with rate μ_i. The only other way to leave state D is for one of the idle nodes that is not a neighbor of any $i \epsilon D$ to begin to transmit. This occurs with rate g_j. Let N_D be the set of all neighbors of all nodes in D. Then the transition from D to {D+j}, $j \epsilon N_D$, occurs with rate g_j. The global balance equations for this system are

$$(\sum_{i \epsilon D} \mu_i + \sum_{j \notin N_D} g_j) Q(D) = \sum_{i \epsilon D} g_i \, Q(D-i) + \sum_{j \notin N_D} \mu_j \, Q(D+j) \qquad (7)$$

where D is one of the special sets defined above.

It is easy to see that these equations are satisfied by

$$Q(D) = \frac{g_i}{\mu_i} Q(D-i) = G_i Q(D-i), \; i \epsilon D \qquad (8)$$

Thus

$$Q(D) = \left(\prod_{i \in D} G_i\right) Q(\phi) \tag{9}$$

where we adopt the convention that $\prod_{i \in \phi} G_i = 1$. Summing over all D, we get

$$\sum_{all\ D} Q(D) = \sum_{all\ D} \left[\prod_{i \in D} Q(\phi)\right] = 1 \tag{10}$$

In the previous section we found we were interested in quantities like P(A), where A is any set of nodes, and P(A) is the probability that all nodes in A are idle, and all nodes not in A may or may not be idle. This can be found by summing Q(D) over all sets D that do not contain nodes in A. Thus

$$P(A) = \sum_{D \subset A^c} Q(D) = \frac{\sum_{D \subset A^c} \left(\prod_{i \in D} G_i\right)}{\sum_{D \subset N} \left(\prod_{i \in D} G_i\right)} \tag{11}$$

where $D \subset A^c$ refers to all such sets contained in the complement of A and N is the set of all nodes. We adopt the shorthand notation.

$$SP(B) = \sum_{D \subset B} \left(\prod_{i \in D} G_i\right) \tag{12}$$

where SP refers to sum of products. Thus

$$P(A) = SP(A^c)/SP(N) \tag{13}$$

Equations (5), (6), and (11) can be used for any topology to generate the solution to our problem. The equations relating the S_{i,n_i}, G_{i,n_i}, and G_j can be solved iteratively. For example, equation (5) now becomes

$$\frac{S_{i,N_i}}{G_{i,N_i}} = \frac{SP([N_i+N_{a_i}]^c)}{SP(N)} \tag{14}$$

where by A+B we mean the union of A and B.

Evaluation of sums of products in equation (12) are made easier by the following two rules. Consider two sets of nodes A and B such that no node in A can hear any node of B. Then

$$SP(A+B) = SP(A)\,SP(B),\ A \bigcap B = \phi \tag{15}$$

Also,

$$SP(A) = SP(A-i) + G_i SP(A-N_i),\ i \epsilon A \tag{16}$$

To prove these rules just consider all products. We have successfully evaluated many complex topologies with these procedures.

There are other relations which will be found useful in extending our model to more complex situations. We prove some of these below. Let C be a cut, i.e., a set of nodes that divides the network into three parts A, B, and C, where A and B have no neighbors in common as in equation (15). Let $A=A_1+A_2$, $B = B_1+B_2$ where $A_1A_2 = B_1B_2 = \phi$. Then

$$P(A_1|C,B_1) = \frac{P(A_1,C,B_1)}{P(C,B_1)} = \frac{SP(A_2+B_2)}{SP(A+B_2)} = \frac{SP(A_2)}{SP(A)}$$

But

$$P(A_1|C) = \frac{P(A_1,C)}{P(C)} = \frac{SP(A_2+B)}{SP(A+B)} = \frac{SP(A_2)}{SP(A)}$$

Thus

$$P(A_1|C,B_1) = P(A_1|C),\ C\ a\ cut \tag{17}$$

we also have

$$P(A_1,B_1|C) = P(A_1|C)P(B_1|C),\ C\ a\ cut \tag{18}$$

In particular if $C = N_i^*$, then

$$P(i|N_i^*,B) = P(i|N_i^*),\ B \subset N_i^c \tag{19}$$

V. A GENERAL CLASS OF PACKET LENGTH DISTRIBUTIONS

In this section we extend the results just proven to include a general class of distributions for the packet length. We will show that the procedures developed for perfect capture are independent of packet length distribution. To prove this we start with a simpler extension. In the above we assumed that all packet lengths are exponentially distributed and have the same mean when transmitted by a node to any of its neighbors. Different nodes may transmit different average length packets, however. Now assume that while all packet lengths are still exponentially distributed, the average length packet transmitted from a node may be different to each of its neighbors. This will be useful in analyzing different protocols (to be presented in a sequel paper) but is presented here as the first step in the desired extension.

Now the state of the network depends upon who is transmitting and to whom. We can keep the same structure by breaking every node into a set of "micronodes", one for each neighbor. These nodes may be indexed by (i,n_i). If i is transmitting to n_i then this node is active, otherwise it is idle. Micronodes are connected in our topology if they can hear each other. Since CSMA still prevails, all micronodes for a given node are fully connected. Furthermore all micronodes for nodes that are connected in the original topology are also fully connected. The analysis now proceeds as above since the Markovian property has been maintained. For example, equation (14) still holds, but now N_i and N_{n_i} are collections of

micronodes. Note that N_i contains the full set of micronodes for i and all neighbors of i. S_{i,n_i} and G_{i,n_i} have the same meaning as before. However they are normalized by $1/\mu_{i,n_i}$, the average packet length for packets going from i to n_i. The terms in the sums of products are G_{i,n_i} for the micronodes.

Let A contain full sets of micronodes and include the node i. Then from equation (16)

$$SP(A) = SP[A-(i,n_i)] + G_{i,n_i}SP(A-N_i) \tag{20}$$

Here we have used the fact that $N_{(i,n_i)} = N_i$, and the notation that N_i is the set of all micronodes of i and neighbors of i. Since this is equivalent to the original set of node neighbors of i we keep the same notation. Repeating for all neighbors of i we get,

$$SP(A) = SP(A-i) + (\sum_{n_i} G_{i,n_i})SP(A-N_i) \tag{21}$$

If we let

$$G_i = \frac{g_i}{\mu_i} = \sum_{n_i} G_{i,n_i} = \sum_{n_i} g_{i,n_i}/\mu_{i,n_i} \tag{22}$$

then equation (16) is preserved. In a similar manner all previously derived equations can be maintained where G_i takes on the definition in equation (22). Here $1/\mu_i$ is an average packet length, averaged over the different average packet lengths to different neighbors in proportion to their scheduled rate.

We now prove our main theorem for this section. Assume that the length of packets transmitted from i to n_i has the density

$$f_{i,n_i}(x) = \sum_j a_{i,n_i,j} \cdot \mu_{i,n_i,j} \cdot e^{-\mu_{i,n_i,j}x} \tag{23}$$

where

$$a_{i,n_i,j} \geq 0 \text{ and } \sum_j a_{i,n_i,j} = 1$$

Thus the length is distributed as a positive sum of exponentials. Another way of looking at this is that the a's are the probabilities of choosing the associated exponential density. Now create micronodes for each triple (i,n_i,j). Here we use $S_{i,n_i,j} = s_{i,n_i,j}/\mu_{i,n_i,j}$ *where* $s_{i,n_i,j} = a_{i,n_i,j}S_{i,n_i}$. The micronodes for some i and any $n_{i,j}$ are fully connected as are the micronodes for neighboring nodes. Equations (20) and (21) now become

$$SP(A) = SP[A - (i,n_i,j)] + G_{i,n_i,j}SP(A-N_i) \tag{24}$$

and

$$SP(A) = SP(A-i) + \left[\sum_{n_i,j} G_{i,n_i,j}\right]SP(A-N_i) \tag{25}$$

In the same manner as above, we let

$$G_i = \frac{g_i}{\mu_i} = \sum_{n_i,j} a_{i,n_i,j} = \sum_{n_i,j} \frac{g_{i,n_i,j}}{\mu_{i,n_i,j}} = \sum_{n_i} \left[\sum_j \frac{g_{i,n_i,j}}{\mu_{i,n_i,j}}\right]$$

$$= \sum_{n_i} G_{i,n_i}.$$

These equations are used to find P(A) in terms of G_i and are identical in form to those derived for exponentially distributed packet lengths.

Equation (5) in turn comes from

$$S_{i,n_i,j} = G_{i,n_i,j} P(N_i - N_{n_i}). \tag{27}$$

Summing over j, we get equation (5)

$$S_{i,n_i} = \sum_j S_{i,n_i,j} = G_{i,n_i} P(N_i - N_{n_i}) \tag{28}$$

Thus all relations between the G's and the S's are preserved. The actual nature of the problem is taken into account by the relationship between the S's and the normalization by the μ's.

We can now restate the above theorem. Let i be any node and $j \epsilon N_i^*$, the neighborhood of i. Let $s_{i,j}$ be the successful (desired) rate from i to j, in packets/sec. Let the density of the packet lengths be

$$f_{i,j}(x) = \sum_k a_{i,j,k} \mu_{i,j,k} \exp^{-\mu_{i,j,k} x}, \text{where } \mu_{i,j,k} > 0,\ a_{i,j,k} \geq 0,$$

$$\text{and } \sum_k a_{i,j,k} = 1.$$

Let $1/\mu_{i,j} = \sum_k a_{i,j,k}/\mu_{i,j,k}$ be the average packet length, in seconds. Let $S_{i,j} = s_{i,j}/\mu_{i,j}$. Then

$$\frac{S_{i,j}}{G_{i,j}} = P(N_i \cup N_j) = a \text{ function of } (G_1, G_2, \cdots)$$

where

$$G_i = \sum_{j \epsilon N_i^*} G_{i,j}.$$

Proof:

let $s_{i,j,k} = a_{i,j,k} s_{i,j}$.

Then $s_{i,j,k} = a_{i,j,k} s_{i,j} = P(N_i \cup N_j) g_{i,j,k}$

where $G_i = \sum_{j \in N_i^*} G_{i,j} = \sum_{j \in N_i^*} \sum_k g_{i,j,k}/\mu_{i,j,k}$.

Dividing $s_{i,j,k}$ by $\mu_{i,j,k}$ and summing over k, we get

$$S_{i,j} = P(N_i \cup N_j) G_{i,j}$$

VI. EXAMPLES OF THE PROCEDURE

As an example consider the chain of four nodes shown in Figure 2. We assume $S_{12} = S_{21} = S_{23} = S_{32} = S_{34} = S_{43} = S$ for simplicity, and perfect capture. Also note that $G_{12} = G_1$, $G_{43} = G_4$ and by symmetry $G_1 = G_4$, $G_{21} = G_{34}$, and $G_{23} = G_{32}$. Also from equation (6), $G_2 = G_{21} + G_{23} = G_3$. From equation (5), we have

$$\frac{S}{G_1} = P(1,2,3) = \frac{S}{G_{21}}, \frac{S}{G_{23}} = P(1,2,3,4)$$

But

$$SP(N) = \sum_{i \in D} (\Pi G_i) = 1 + G_1 + G_2 + G_3 + G_4 + G_1G_3 + G_2G_4 + G_1G_4$$

$$= 1 + 2G_1 + 2G_2 + 2G_1G_2 + G_1^2 = \Delta$$

and

$$P(1,2,3,4) = 1/\Delta,\ P(1,2,3) = (1+G_4)/\Delta = (1+G_1)/\Delta$$

Solving, we get

$$G_2 = G_1(2+G_1) \text{ and } S = G_1(1+G_1)/\Delta\ .$$

or

$$S = G_1(1+G_1)/(1+6G_1+7G_1^2+2G_1^3).$$

We can now find the maximum value of S possible, the throughput of the chain, which is .128, obtained when $G_1 = 0.71$.

In general the equations cannot be solved as simply as for this four node example. Equation (11) is used to get expressions for P(A) in terms of G_l. Then equation (14) is used to get expressions for S_{l,s_l} in terms of G_{l,s_l} and P(A). Equation (6) provides the relation between G_l and G_{l,s_l}. The S_{l,s_l} are found from the offered traffic, the routing, and other assumptions and are considered as inputs. The equations are iterated until a solution of G's for a set of S's is found. The maximum set of S's possible is considered the throughput, or capacity, of the network. For some modest size problems, as above, the equations can be solved directly.

As a second example consider the star topology shown in Figure 5. Here assume there are L legs of N = 2 nodes each. Denote the center node by 0, the nodes one hop out by 1, and the other nodes by 2. Further assume symmetrical traffic in the nodes and $S_{01} = S_{10} = S_{12} = S_{21} = S$. Then LS is the total traffic successfully transmitted by node 0. The equations are

$$\frac{S_{01}}{G_{01}} = \frac{LS}{G_0} = \frac{S_{10}}{G_{10}} = \frac{S}{G_{10}} = P(\textit{all but } L-1 \neq 2 \textit{ nodes}) = \frac{SP(L-1 \neq 2 \textit{ nodes})}{SP(N)}$$

$$= (1+G_2)^{L-1}/\Delta$$

$$\frac{S_{12}}{G_{12}} = \frac{S}{G_{12}} = \frac{S_{21}}{G_{21}} = \frac{S}{G_2} = P(0,1,2) = \frac{SP(L-1 \textit{ legs})}{SP(N)} = (1+G_1+G_2)^{L-1}/\Delta$$

where

$$\Delta = SP(N) = (1+G_1+G_2)^L + G_0(1+G_2)^L$$

$$G_1 = G_{10} + G_{12}$$

But $G_{10} = G_0/L$ and $G_{12} = G_2$, so $G_1 = G_2 + G_0/L$. Thus we have two equations:

$$LS/G_0 = (1+G_2)^{L-1}/\Delta \text{ and } S/G_2 = (1-2G_2+G_0/L)^{L-1}/\Delta$$

For any $S < S_{max}$, they can be solved for G_0 and G_2. Alternatively for any G_0 we can find the corresponding G_2 by solving

$$G_0(1+G_2)^{L-1} = G_2(1-2G_2+G_0/L)^{L-1}.$$

Then the relation between S and G_2 can be studied. We then find the maximum S, S_{max}, possible. LS_{max} is the maximum throughput of the star.

For larger problems we will get several equations of the form

$$G_0 = LS\Delta/(1+G_2)^{L-1}$$
$$G_2 = S\Delta/(1+2G_2+G_0/L)^{L-1}$$

For any S we solve these iteratively. Since $G_0 \geq LS$, $G_1 \geq 2S$, and $G_2 \geq S$, the lower bounds are good starting points for G_i. For S sufficiently less than S_{max} we have found the iteration converges monotonically and rapidly. As S approaches S_{max} from below the convergence is still monotonic but slows appreciably. For $S > S_{max}$ the iteration does not converge and often diverges dramatically. We have uncovered no serious numerical problems with this procedure in the many examples we have evaluated.

VII. NUMERICAL EXAMPLES

We consider here three different topological structures with exponentially distributed packet lengths and perfect capture. We assume all $S_{i,R_i} = S$, for all i, and take full advantage of symmetry. The three topologies, shown in Figures 3, 4, and 5, are a chain, a ring, and a star, all with various lengths. In each case we find the maximum throughput, S. These are given in Table I.

The maximum one way throughput for a long chain is S = .086. This throughput is approached when the lengths of chains exceed 10. For smaller length chains the throughput is

higher. In CSMA transmissions of neighbors may not overlap in time. Since each nodes transmits successfully 2S packets per average packet transmission time, then we must have $S \leq 1/5$. The throughput is slightly smaller than half of this limit. The cause of the reduction is collisions from transmissions two hops away, the so-called "hidden terminals". The throughput of S = .086 for a chain, although the maximum possible, is not a useful operating point. As in ALOHA, this is the point at which delays become infinite and the system is unstable. The network would have to be operated at some lower level.

It is instructive to compare the performance of multihop CSMA with that of slotted ALOHA. Let p be the probability of transmission in one direction at a node. Then $S = p(1-2p)^2$ for a long chain. S_{max} here is .074 which is approximately 14% less than that for CSMA. There are two factors working here. CSMA will produce less collisions since neighbors will not interfere with each other. Hidden terminals will still produce collisions. (All terminals are hidden in ALOHA). But CSMA prohibits possible successful transmissions. For instance, node 3 can transmit successfully to node 2 while node 4 is transmitting successfully to node 5. This is possible in ALOHA but prohibited in CSMA. This is one of the prices paid to control collisions.

We note that for a ring greater than 7 nodes the maximum throughput is the same as that for a long chain. This is expected since the congestion is now in the middle and it is unimportant whether or not the chain is closed. A star with two legs is just a chain with N for the star replaced by 2N+1 for the chain.

Consider the star configuration as representing the center node (0) trying to transmit to some node or nodes far away via many repeaters. For one leg, the maximum rate is .086. For two legs, the maximum rate is 2S or .172, exactly twice. The results are shown in Table II where the throughput of the center node is given by LS. We see from Table II that whereas the throughput doubles for L = 2, it increases only by 20% when L=3, by 4-5% further when L=4, and by 2% when L=5. Congestion at the central node is limiting its ability to increase its

throughput. Additional legs, beyond three, are not really helpful.

We have investigated ways of reducing the congestion at the center node. For larger L, the traffic in each leg is limited by congestion at the center. The collisions that cause most problems are for transmissions from the first level of nodes (1, N+1, 2N+1,..., (L-1)N+1) to the center. These are collided with by other first level nodes. To reduce these collisions we considered connecting the first level nodes in a ring and then fully connecting them. These results are also summarized in Table I. When the first level is unconnected the throughput saturates at .229. When the first level is fully connected the throughput with 9 legs is .252, a 15% increase. For four legs, the ring connected topology is best, providing some compromise between reducing collisions and allowing simultaneous transmissions.

The best that we can expect in the fully connected case is $LS \leq 1/3$. This is because all transmissions from the center node and all first level nodes cannot overlap. We will discuss asymptotic results with even larger stars and chains in the next section.

VIII. ASYMPTOTIC RESULTS

We are interested in asymptotic results for several reasons. They provide us with the limiting behavior of the finite networks previously studied. Since the behavior of these networks seems to converge rapidly with their size, if asymptotic results are easier to obtain, they would be useful. We are also interested in very large networks. A final reason is to verify some of the bounding arguments on throughput made in the last section.

We first consider an infinitely long chain. We let $S_{i,j+1} = S_{i,j-1} = S$. Then all nodes are identical. Also $G_{i,j+1} = G_{i,j-1} = G_i/2$. Thus with $G_i = G$,

$$\frac{2S}{G_i} = \frac{2S}{G} = P(i-1, i, i+1, i+2) = \frac{SP(-\infty,\ldots,i-2)SP(i+3,\ldots,\infty)}{SP(-\infty,\ldots,\infty)} \tag{29}$$

We can write the denominator as

$SP(-\infty,\ldots,\infty) = SP(-\infty,\cdots i-1)SP(i+1,\ldots,\infty) + G_i SP(-\infty,\ldots,i-2)SP(i+2,\ldots,\infty)$.

Now let $Q_k = \frac{SP(i-k,\ldots,\infty)}{SP(i,\ldots,\infty)} = \frac{SP(-\infty,\ldots,i+k)}{SP(-\infty,\ldots,i)}$. We observe that $Q_k \geq 1$ for $k \geq 0$ and if it converges is independent of i. Then equation (29) becomes

$$\frac{2S}{G} = \frac{1}{Q_1 Q_2 + G Q_1} \tag{30}$$

But

$$Q_k = \frac{SP(-\infty,\ldots,i+k-1) + G\ SP(-\infty,\ldots,i+k-2)}{SP(-\infty,\ldots,i)} = Q_{k-1} + G Q_{k-2} \tag{31}$$

and $Q_0 = 1$. Thus $Q_2 = Q_1 + G$. Therefore

$$2S/G = \frac{1}{Q_1(Q_1 + 2G)}$$

or

$$S = \frac{G}{2Q_1(Q_1+2G)} \tag{32}$$

We note that since $Q_{-1} = \frac{1}{Q_1}$, from equation (31) we have

$$Q_1 = Q_0 + \frac{G}{Q_1} = 1 + \frac{G}{Q_1} \quad or$$

$$G = Q_1(Q_1-1) \tag{33}$$

Finally we have

$$S = \frac{Q_1-1}{2Q_1(2Q_1-1)}, Q_1 \geq 1 \tag{34}$$

The maximum value of S is .086, reached when $Q_1 = 1.7$ *or* $G = 1.2$.

IX. EXTENSIONS AND CONCLUSIONS

We have presented a simple but fairly realistic model of a multihop packet radio network and have obtained maximum throughputs for general topologies and packet lengths. We have assumed perfect reception of acknowledgments and have not included additional traffic due to end-to-end acknowledgments. Some aspects of acknowledgments can be included by increasing the required traffic. We are investigating the effect of imperfect acknowledgments and different retransmission strategies. The model should still be useful under these extensions.

REFERENCES

[1] L. Kleinrock and F. Tobagi, "Packet switching in radio channels," parts I and II, IEEE Trans. Commun., vol. COM-23, pp. 1400-1433, December 1975.

[2] F. Tobagi, "Analysis of a Two-Hop Centralized Packet Radio Network", Part II: CSMA IEEE Trans. Commun., COM-28, pp. 208-216, February 1980.

[3] R. E. Kahn, "The organization of computer resources into a packet radio network," IEEE Trans. Communications, Vol. COM-25, pp. 169-178, January 1977.

[4] I. Gitman, R. Van Slyke, and H. Frank, "Routing in packet-switching broadcast radio networks," IEEE Trans. Communications, vol. COM-24, pp. 926-930, September 1976.

[5] R. Boorstyn and A. Kershenbaum, "Throughput Analysis of Multi-hop Packet Radio Networks," IEEE-ICC'80, pp. 13.6.1-13.6.6, Seattle, WA., June 1980.

[6] V. Sahin, "Analysis of Multihop Packet Radio Networks", Ph.D Thesis, Polytechnic Institute of New York, June 1982.

Table I

Maximum One-Way Throughput (S)

Number of Nodes, N	Chain	Ring	Star Number of Legs, L L=2	L=3	L=4	L=5
1	1.000	1.000	.167	.103	.074	.058
2	.500	.500	.111	.076	.057	
3	.167	.167	.097	.072	.055	
4	.128	.073	.092	.070	.054	
5	.111	.100		.069	.054	.044
6	.102	.083				
7	.097	.087				
8	.094	.085				
9	.092	.086				
10	.091	.086				
∞	.086	.086	.086	.069	.054	.044

Table II

Maximum One-Way Total Throughput of the

Central Node in a Large Star Arrangement

	Maximum Throughput, LS Center Arrangement		
Number of Legs	*Unconnected*	*Ring-Connected*	*Fully Connected*
1	.086		
2	.172		
3	.207	.198	.198
4	.216	.228	.216
5	.220	.230	.230
6	.220		.240
7			.245
8			.248
9			.252

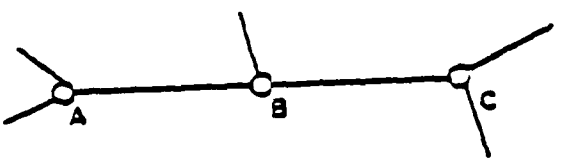


Figure 1. A part of a network

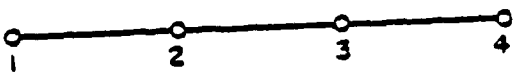


Figure 2. A four node chain

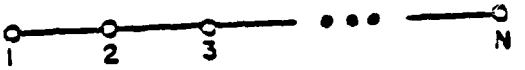


Figure 3. A chain

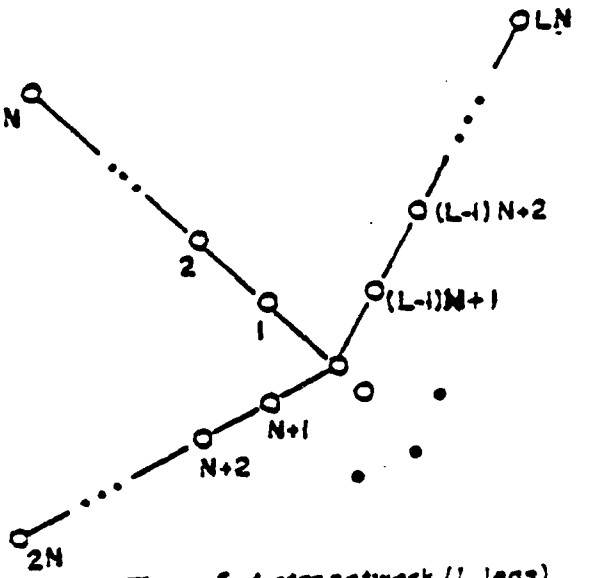


Figure 5. A star network (L legs)

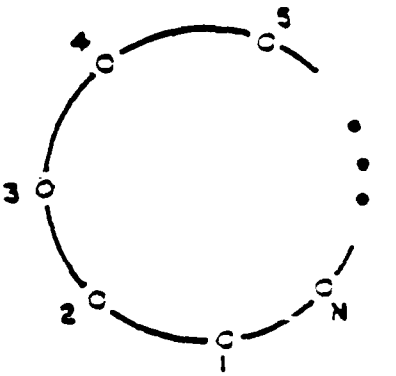


Figure 4. A ring

APPENDIX D

To appear in Networks Journal.

Generalized Augmenting Paths for the Solution of Combinatorial Optimization Problems

Aaron Kershenbaum
Polytechnic Institute of New York

Abstract

Alternating chain procedures can be thought of as generalizations of the greedy algorithm in that instead of accepting the best remaining element, they seek to obtain a better augmentation by examining a wider range of alternatives. It is possible to generalize the notion of an augmenting sequence to include augmentations which are in effect trees as opposed to simply paths such that these augmentations are sufficient to guarantee optimality. Unfortunately, in the worst case, these trees are of exponential size. We examine the application of such generalized augmenting sequences to the solution of NP-complete problems and examine their effectiveness and efficiency.

Introduction

The theory of NP-completeness, which was first expounded by Cook [1], has led to a search for a unified treatment of combinatorial optimization problems. Cook was able to characterize a very large class of interesting and important problems as being equivalent in the sense that an efficient algorithm capable of finding an optimal solution to any one of these problems can be used to obtain optimal solutions to all of the others. Many papers by many authors and an excellent compendium [2] of problems in this class (as well as techniques for proving that a problem is in this class) have been published since Cook's seminal paper. Problems in this class are called NP-complete problems (or, more properly, NP-hard when they are optimization problems as opposed to decision problems).

Cook's results can be interpreted in several ways. One of these is to say that many clever people have spent many years trying and failing to find efficient algorithms for individual problems in this class. Surely one of them would have succeeded if, in fact, such algorithms existed. Hence, it is unlikely that such an algorithm will be found and it is tempting to stop looking for one. This leads to the development of heuristics for the solution of such problems [3] and to probabilistic methods [4].

An alternate interpretation is that this pessimistic view is justified only with respect to algorithms which guarantee optimal solutions and reasonable runtimes for all instances (input data sets) of a problem. In this paper we speak of an algorithm's runtime being reasonable if it grows polynomially rather than exponentially with the size of the problem. This does not preclude the existence of algorithms with guaranteed reasonable runtimes and which yield optimal (or near-optimal) solutions with high probability. Nor does it preclude the existence of algorithms

which guarantee optimal solutions and which have reasonable runtimes with high probability. There are many examples of both types of algorithms which are used in practice to solve specific NP-complete problems. Most important, the theory of NP-completeness does not preclude or even lessen the likelihood of the existence of algorithms which solve specific (nontrivial) instances of a problem and guarantee both an optimal solution and reasonable runtime.

In this paper, we explore this second, more optimistic, point of view and present a family of algorithms for the solution of an NP-complete problem. Some algorithms in this family have guaranteed reasonable runtimes. Others guarantee optimal solutions. While the algorithms are presented for the solution of a specific problem, the technique can be extended to the solution of other problems as well.

Matroid Theory

A specific way of approaching the solution of many combinatorial optimization problems is via matroid theory. The excellent book by Lawler [5] gives a complete treatment of this. Here we outline the fundamentals of this theory which are necessary for the presentation which follows.

A matroid is a couple (E,F) where E is a finite set of m elements:

$$E = \{e_j \mid j = 1,2, \ldots M\}$$

and F is a family of independent subsets of E. The notion of independence is quite general. We require, however, that it satisfy two properties:

P1: Every subset of an independent set is independent, i.e., if

$I \in F$ and $J \subset I$ then $J \in F$

P2: If I_P and I_{P+1} are independent subsets of E containing P and P + 1 elements, respectively, then there exists an element, $e \in I_{P+1}$ ($e \notin I_P$) such that $I_P \cup \{e\}$ is an independent set containing P + 1 elements.

Given two matroids, (E,F_1) and (E,F_2), defined on the same set of elements, but using two different notions of independence, we define an intersection of them to be any subset, $I \subset E$, such that $I \in F_1$, and $I \in F_2$. This definition can be extended to cover three or more matroids as well.

Many combinatorial optimization problems can be thought of as finding the best independent set in a matroid or the best intersection of two or more matroids. If weights, w_j, are associated with the elements, e_j, in E, then one can speak of the best set as being the one with largest total weight. The maximal (or minimal) spanning tree problem can be thought of as finding the maximum (or minimum) weight independent set in a matroid (E,F) where E is the set of edges in the graph and F is the family of forests. A forest is defined to be a set of 0 or more

edges which do not contain a circuit. As another example, Lawler [5, p. 304] shows that the Traveling Salesman Problem can be thought of as finding the best intersection of three matroids. The problem of finding the maximum weight intersection of three matroids has been shown to be NP-complete [2]. Lawler shows [5, p. 364] that the problem of finding intersections of four or more matroids can be reduced to that of finding intersections of three. There are many other combinatorial optimization problems which can be naturally thought of as matroid intersection problems. The theory of NP-completeness assures us that all problems can be thought if in this way.

We will consider one of the simplest possible 3-Matroid Intersection Problems in the sequel for the sake of clarity. The problem considered is the Three Dimensional Assignment Problem (TDAP). In this problem, we are given N people, N jobs, and N days. There is a cost, C_{ijk} of having person i doing job j on day k. Each person is to do only one job, each job is to be done only once, and only one job is to be done on a day. Formally the problem is:

$$\text{Minimize} \quad Z = \sum_{i,j,k} C_{ijk} X_{ijk}$$

such that

$$\sum_{i,j} X_{ijk} = \sum_{i,k} X_{ijk} = \sum_{j,k} X_{ijk} = 1 \text{ for } i,j,k = 1,2, \ldots N \qquad X_{ijk} \in \{0,1\}$$

Thus, setting X_{ijk} to 1 corresponds to having person i do job j on day k. This problem can be viewed as an intersection of three partition matroids. Given a set of elements, E (in this case, the X_{ijk}), a partition matroid can be defined by a partition of E and a vector, A, constraining the number of elements of E which may be selected from any part of the partition. Formally, we have the partition of E into subsets E_j, j = 1, ... k, where

$$\bigcup_j E_j = E \qquad \text{and} \qquad E_i \cap E_j = \phi \text{ for } i \neq j$$

and an integer vector $A = \{a_j | \ j = 1, \ldots k\}$

A matroid (E,F) is then defined where F consists of all subsets, I, of E formed by selecting no more than a_j elements of E_j.

In the case of the TDAP, the first partition of the X_{ijk} is by person, i.e.,

$$E_i = \{X_{ijk} | j = 1, \ldots N; \ k = 1, \ldots N\}$$

and $a_i = 1$ for all i. The independent sets in this first matroid correspond to assigning each person at most one job. Similarly, two more matroids can be defined to constrain jobs and days. Intersections of these three matroids correspond to feasible partial assignments and intersections of maximum cardinality correspond

to feasible complete assignments. If we define weights W_{ijk} associated with the X_{ijk}:

$$W_{ijk} = C - C_{ijk}$$

where C is larger than any C_{ijk}, then the maximum weight intersection corresponds to the optimal solution to the TDAP.

Augmenting Paths

We now define a family of algorithms for the solution of matroid intersection problems. These are generalizations of the basic procedure given in [6].

Given a matroid (E,F) (and hence a notion of independence) and a (not necessarily independent) subset S, of E, we define the span of S, denoted sp(S), as S together with all elements of E not independent of the elements in S, that is

$$sp(S) = \{ e \mid I \cup \{e\} \notin F \text{ where I is any independent subset of S}\}$$

If S is an independent set and $e \in sp(S)$ then e forms a unique cycle, which we denote by C(e), with S. A cycle is a dependent set which becomes independent if any element is removed from it.

If the matroid intersection problem only involves two matroids, we can obtain a maximum weight intersection by producing a sequence of intersections, $I^{(K)}$, containing K elements, for K = 1,2, ... m. Each $I^{(K)}$ is the maximum weight intersection containing K elements. The algorithm which produces the $I^{(K)}$ is called an augmenting path procedure because it augments $I^{(K)}$ to produce $I^{(K+1)}$ by finding the longest path in the graph $G^{(K)}$ defined below.

We define $G^{(K)}$ to be a bipartite graph with nodes corresponding to the elements, e_j, of E plus distinguished start and finish nodes, a and z. Directed arcs are defined as follows:

(a,i)	$i \in E - sp_1(I^{(K)})$	(i,j)	$i \in E - I^{(K)}$, $j \in C^{(2)}(i)$
(i,z)	$i \in E - sp_2(I^{(K)})$	(j,i)	$i \in E - I^{(K)}$, $j \in C^{(1)}(i)$

Paths from a to z correspond to augmentations of $I^{(K)}$, that is, to sets of elements to be added or deleted from $I^{(K)}$ to produce an intersection with K + 1 elements. Notice that all a to z paths go alternately through nodes not contained in $I^{(K)}$ (which are to be added to $I^{(K)}$) and nodes in $I^{(K)}$ (which are to be deleted). Note also that there is one more node of the former type than there is of the latter and hence an augmentation results. If we associate lengths with the arcs equal to the weights of the elements which the nodes correspond to (positive for elements to be added and negative for elements to be deleted), then the length of a path corresponds to the incremental weight of the augmentation. The longest

path results in an optimal augmentation. Such a path can be found using a shortest path algorithm suitably modified to find longest paths. $G^{(K)}$ contains no positive cycles and so the algorithm converges.

These augmentations do, indeed, result in intersections. As one passes through nodes from a to z we see that an element is added preserving independence in the first matroid but not the second. An element is then deleted restoring independence in the second matroid and hence the intersection. A node is then added which, because of the deleted node, maintains independence in the first matroid. This process continues until the added element maintains independence in the second matroid as well as the first, thus completing the augmentation.

As an example, consider a two dimensional assignment problem (involving, say, only people and jobs.) The W_{ij}'s for this problem are given in Figure 1. $I^{(2)}$ is clearly 11,22, i.e., person 1 assigned to job 1, and person 2 assigned to job 2. $G^{(2)}$ is shown in Figure 2. The arc lengths are shown as are the lengths of the longest paths to each node from node a. The longest a to z path is a,11,12,22,23,z which corresponds to deleting 11 and 22 from the intersection and adding 31,12, and 23. The length of this path, 7, is the difference between the weight of $I^{(3)}$ and $I^{(2)}$. A complete description of this process and a proof of its validity is given in [5].

Generalized Augumenting Paths

In the graph shown in Figure 2, one can obtain an optimal augmentation (i.e., one which takes us from an optimal assignment of K elements to an optimal assignment of K + 1) because:

1. If the current intersection is not maximal then an augmenting path exists.
2. The labels given to the nodes during the longest path algorithm completely summarize the augmenting paths.

We now wish to generalize the notion of an augmenting path, and hence the entire procedure, to the problem of the intersection of three matroids. One way of doing this is to "freeze" one of the matroids and only consider alternating sequences within the other two. In this case the first node, s_1 in an augmenting path would be independent of $I^{(K)}$ in two of the three matroids (or in all three, in which case it is the only node in the augmenting path). Say s_1 is independent of $I^{(K)}$ in the first and third matroids. We could then freeze the third matroid and maintain the same span within the third matroid throughout the augmenting path. Thus, the deletion of s_i for i even reduces this span and the addition of s_i for i odd restores it. We thus reduce the search space to two matroids and the same polynomial bounded procedure will work. Note that, alternatively, we could have considered the first matroid frozen. Indeed, it is so frozen in the two matroid intersection algorithm. Thus, there are three types of augmenting paths, one for each matroid

within which s_1 is dependent. Unfortunately, while this procedure is polynomial bounded, it does not guarantee optimal solutions as there are augmentations which have no such corresponding argumenting path.

In order to guarantee that all augmentations are explored, we must relax the definition of an augmenting path still further to include cases where independence is not necessarily restored by the deletion of s_i for i even. Thus, an augmenting path may start with any element, s_1, which is independent of $I^{(K)}$ in at least one of the matroids. Unlike the procedure given for two matroids, one may begin with independence in any matroid. Consider the graph shown in Figure 3 corresponding to two augmenting paths, Path 1 and Path 2, for the partial assignment 111,222,333 (i.e., person 1 to job 1 on day 1, etc.) in a TDAP. These paths are not strictly comparable in that they exclude different elements along the way. Thus in Figure 2, when node 22 is labeled using the path a,32,22 it is equivalent (in terms of how the path can continue, not necessarily in terms of the numerical value of the label) to being labeled using the path a,31,11,12,22. In Figure 3, however, when node 111 is labeled using the path a,411,111 it is different from labeling 111 using the path a,154,111 because different continuations of these paths are possible. Thus starting with a,411,111 we can continue to 152 but not 215 and, conversely, starting with a,154,111 we can continue with 215 but not 152. Thus, Path 1 and Path 2 are not comparable in terms of their lengths only.

Such paths must also be compared in terms of their spans. We note that if two paths from a to some node i result in sets having identical spans then the same continuations of both paths are possible. (This was the case for intersections of two matroids.) Indeed, it is possible for paths to have slightly different spans and still have the same set of possible continuations. In particular, if the only difference in the intersections of the spans of two paths are nodes outside the intersection of the spans of $I^{(K)}$, then the paths are comparable. We can thus generalize the augmenting path procedure to consider all undominated a to z paths where one path dominates another only if it has the same continuations and a larger length.

The notion of a path itself, however, must be generalized as well. In the case of 3 matroids, not all augmentations correspond to paths. We see an example of this for a TDAP. The augmentation [412,234,341,123] - [111,222,333] does not correspond to any path in the conventional sense. It is possible however, to extend the augmenting path procedure to include such augmentations by extending the notion of a path.

We define a generalized augmenting path with respect to an intersection $I^{(K)}$ to be a sequence of nodes $S = (s_1, s_2, \ldots s_m)$ where $s_i \in E - I^{(K)}$ for odd i and $s_i \in I^{(K)}$ for even i. As before, $I^{(K)} + s_1 - s_2 + s_3 - \ldots + s_m$ is an intersection. Also, the even s_i are deleted in order to remove dependencies created by the inclusion

of the odd s_i. Now, however, the subsequences $I^{(K)} + s_1 - s_2 + \ldots - s_j$ for even j need not correspond to intersections.

One can thus guarantee an optimal intersection as in the case of two matroids. The number of generalized augmenting paths one may need to consider, however, may grow exponentially with K. In practice, however, the number of such paths can be controlled at the expense of optimality. First, the length of any path, (s_1, s_2, ..., s_j), can be reduced by a penalty to account for the nodes which still must be deleted to restore the intersection. In the case of arbitrary matroids, this may be complex to compute. In the case of the TDAP, however, where 3 partition matroids are involved, and all cycles contain 2 elements, it is easily computed.

In some cases the above may keep the computations reasonable. In others, it may be necessary to reduce the number of paths considered by relaxing the definition of dominance. This will also result in a heuristic rather than an optimal solution. In the case of the TDAP, one such relaxation is to ignore differences in the spans outside the intersection of the span of $I^{(K)}$. This is motivated by the fact that we consider deleting elements in $I^{(K)}$ in order to include elements blocked by them.

We can thus consider a hierarchy of generalized augmenting path procedures with *increasingly stringent dominance criteria and increasing runtime*. A tradeoff between optimality and runtime is then available. We are currently investigating this tradeoff using the TDAP as an example.

References

1. Cook. S.A., "The Complexity of Theorem-Proving Procedures," Proc. of Third Ann. ACM Symposium on Theory of Computing, 1971, p 151-158.
2. Garey, M.R. and D.S. Johnson, Computers & Intractability, W.H. Freeman, 1979.
3. Sahni, S. and E. Horowitz, "Combinatorial Problems: Reducibility and Approximation," Operations Research 26(4), 1978.
4. Karp, R.M., "The Probabalistic Analysis of Some Combinatorial Search Algorithms," in Algorithms and Complexity, Academic Press 1976.
5. Lawler, E., Combinatorial Optimization: Networks and Matroids, Holt, Rinehart & Winston, 1976.
6. Edmonds, J., "Matroid Intersection," in Discrete Optimization I, North Holland Publishers Co. p. 39-49.

Person \ Job	1	2	3
1	10	9	5
2	5	10	9
3	9	5	1

FIGURE 1 - Cost Matrix

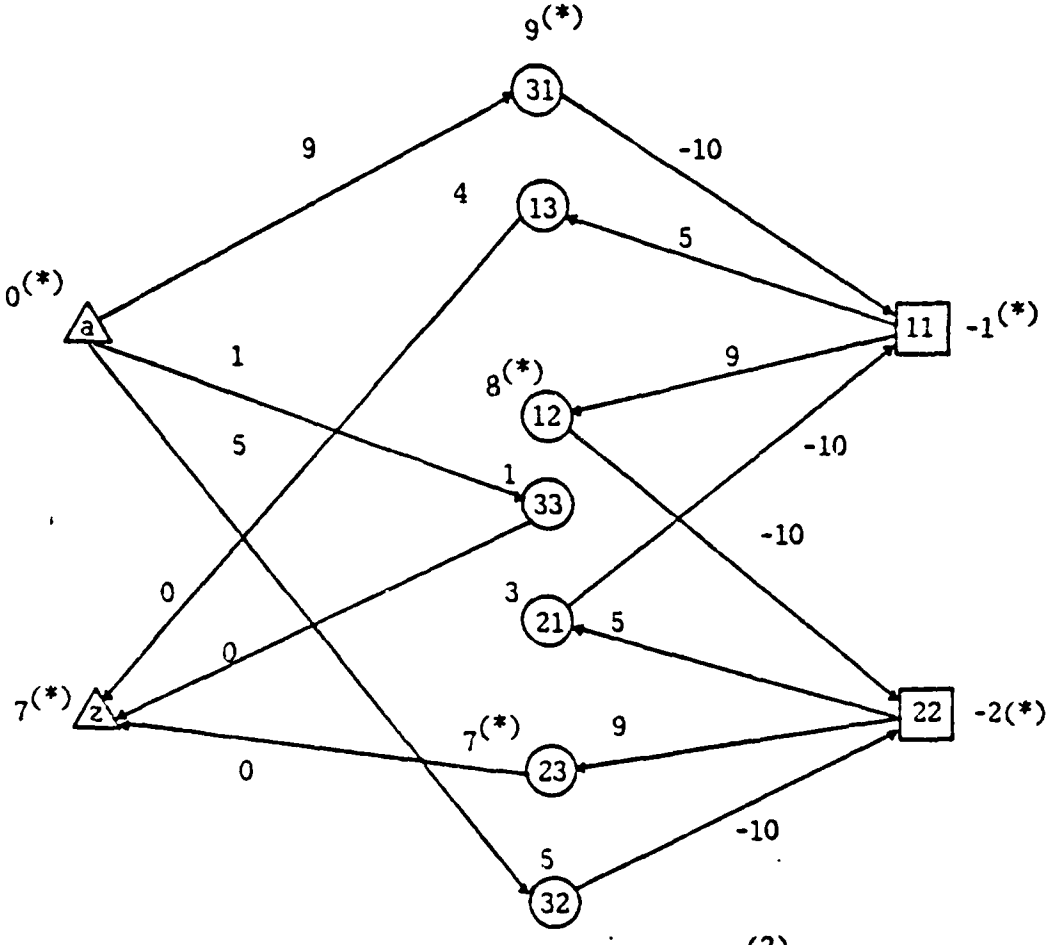


FIGURE 2 - Bipartite Graph $G^{(2)}$

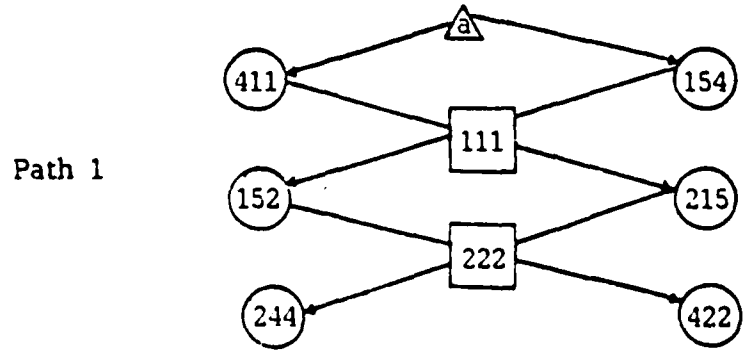


FIGURE 3

APPENDIX E

Centralized Teleprocessing Network Design

By

Aaron Kershenbaum

and

Robert R. Boorstyn
Polytechnic Institute of New York

Abstract

The problem considered is that of finding an optimal (minimum cost) design for a centralized processing network given a set of locations, traffic magnitudes between these locations, and a single common source or destination. Several heuristics, which are efficient (in terms of their execution time and memory requirements on a digital computer) and which produce seemingly good results, have already been developed and are currently accepted techniques. Some work has also been done on finding optimal solutions to this problem both as a design tool and as a means of verifying the effectiveness of proposed heuristics. We focus in this latter area. Currently known techniques for the optimal solution of this problem via integer programming have fallen short of the desired objectives as they require too much memory and running time to be able to treat problems of realistic size and complexity. We develop an improved technique which is capable of handing more realistic problems.

This work was supported in part by the U.S. Army CORADCOM, Contract No. DAAK 80-80-K-0579 , and by the National Science Foundation, Grant No. ENG-7908120

1. INTRODUCTION AND PROBLEM STATEMENT

The problem considered is that of finding an optimal (minimum cost) design for a centralized telecommunication network given a set of locations, traffic magnitudes between these locations, and a single common source or destination. The vast majority of telecommunication networks currently in existence are of this type. Thus, this problem has been much studied (2,3,4,5,8,9,12,16,24,28,31,32).

Several heuristics, which are efficient (in terms of their execution time and memory requirements on a digital computer) and which produce seemingly good results, have already been developed and are currently accepted techniques. Some work has also been done on finding optimal solutions to this problem as a means of verifying the effectiveness of proposed heuristics. Currently know techniques for the optimal solution to this problem via integer programming have fallen short of the desired objective as they require too much memory and running time to be able to treat problems of realistic size and complexity. We develop an improved technique which is capable of handling problems of realistic size.

More formally, the problem considered here is that of finding a minimum spanning tree subject to one or more constraints which in general are equivalent to demanding that the sum of the traffic associated with the nodes in any subtree must not exceed some predetermined maximum.

A minimum spanning tree is a loop-free collection of arcs joining a set of nodes such that the sum of the lengths of the arcs is minimal. In the case of a communication network, these collections of arcs are called multidrop lines.

It should be noted that this constraint form is quite general and encompasses many real-world constraints which arise in the design of centralized telecommunications networks. Thus, for example, in addition to treating the obvious constraint imposed by line capacity, it is possible to treat a restriction on the number of terminals on a multidrop line by associating a uniform traffic with each terminal. Also, the length (cost) functions which can be treated are quite general. Any function which is not a function of the tree chosen is permissable.

Formally, we seek to solve the following problem:

Given

1. A vertex (node) set $V = \{v_i | i=0,1,\ldots,n\}$ representing the terminal locations in the network. Node v_0 is a distinguished node which we will refer to as the center.
2. A symmetric function giving the length (cost) d_{ij} of an arc between any pair of locations.
3. A constraint, m, on the number of nodes which may share a multidrop line. This constraint can be generalized to allow a weight or traffic, c_i, to be associated with each node and to require that the sum of the weights associated with the nodes on any multidrop line not exceed m.

We define the set of nodes in the j^{th} multidrop line to be V_j and the multidrop line itself to be a minimal spanning tree T_{V_j} on $V_j \cup \{v_0\}$. Thus, the constraint can be stated in terms of the cardinality of V_j as $|V_j| \leq m \ \forall_j$. In the more general form, the constraint would be $\sum_{v_i \in V_j} c_i \leq m \ \forall_j$. We wish to find a tree, T_V^* of minimum total length satisfying the constraint in 3 above. That is, we wish to

minimize $\sum_{i=1}^{N} d_{ip_i}$ subject to 3, where v_{P_i} is the immediate predecessor of v_i, i.e., the node closest to v_i on the path between v_i and v_0 in T, and T is any spanning tree. We consider exact (optimal) solutions to this problem. The primary motivation for the work is to develop an exact algorithm capable of permitting study of the performance of heuristics on a broader class of problems than was previously studied, to gain insight into the performance of both exact and heuristic procedures and, in particular, to pinpoint where and why they fail.

II. OUTLINE OF A NEW OPTIMAL SOLUTION TECHNIQUE

There currenty exist several techniques which will yield optimal solutions to the CMST problem. These techniques can be divided into two classes - branch exchange methods (as proposed by Lin (22) and Frank (9)) and branch and bound methods (10,22). We concentrate on the latter class of techniques.

The specific application of branch and bound techniques to the solution of the CMST problem was proposed by Chandy and Russell (3) and was subsequently refined (2) so that it could treat somewhat more meaningful problems. Subsequently, Elias and Ferguson (4) proposed further refinements and thereby expanded the range of applicability of the technique. Gavish (34) recently developed a bound using Lagrangean relaxation.

The basic technique is, as has already been mentioned, a branch and bound algorithm. The original problem considered has all branches in the category "permissible," i.e., any branch may or may not be part of the final solution. Subproblems are generated by selecting a permissible branch and making it "prohibited" in one subproblem or "required" in another.

The relaxation used is simply to generate a modified MST by including all "required" branches, excluding all "prohibited" branches, and forming the tree of minimum total length by connecting (as yet unconnected) nodes using remaining ("permissible") branches.

Clearly, a solution obtained in this manner is a lower bound on the value of a feasible solution to the subproblem as it is the tree of minimum length. Note also that in the case where all arcs are specified (prohibited or required), the lower bound and solution are

identical and the subproblem fathoms. In general, the subproblem fathoms when

1. No feasible solution exists to the subproblem. This occurs, when the required branches form a loop, when the required branches create a subtree violating the constraints, or when the prohibited branches disconnect the network. Other criteria exist but are difficult to test for.
2. The lower bound equals or exceeds the value of the best solution found thus far.
3. The lower bound solution is feasible.

When all subproblems have fathomed, the current best solution is the global optimum.

A number of observations have been made, which can be used to accelerate a basic branch and bound technique. One of these, which is used in the sequel, is given in Theorem 1 below:

Theorem 1: (3)

If branches (v_0,v_{j_1}), $(v_0,v_{j_2}),\ldots,$ (v_0,v_{j_K}), are part of some MST, T, on V then there exists a CMST including these edges.

Corollary:

If arcs (v_0,v_{j_1}), $(v_0,v_{j_2}),\ldots,$ (v_0,v_{j_K}) are present in the modified MST produced in any subproblem, then (if any CMST's exist in the subproblem) there exists a CMST on the subproblem containing these arcs.

The preceding theorem and corollary allow one to avoid considering subproblems with such arcs prohibited. The techniques developed in the sequel make explicit use of both observations as well as others made in the references cited.

The inherent problem with the existing procedures lies in the relaxation method used. At each step, the problem is relaxed to a modified MST. Unfortunately, this bound is often too loose to eliminate a sufficiently large percentage of the subproblems to make the procedure practical. This is particularly evident when the constraints are tight; it is for such problems that the relaxation is loosest. Unfortunately, it is also for that class of subproblems that the known heuristics display the widest variation in the quality of solutions.

Note that any optimal solution to the CMST problem has the property that all subtrees are MST's on the set of nodes contained in the subtree and the center. Thus, it suffices to find the optimal partition of the nodes into subtrees. The technique which is developed in the following sections will thus generate partitions of the nodes.

The technique works within the framework of branch and bound algorithms, as did the techniques referred to above. We develop two algorithms, one based on generating subproblems by restricting nodes, and the other based on generating subproblems by restricting arcs. These techniques differ from previous ones in that the relaxation used here is tighter and thus, a smaller number of subproblems need be examined.

We begin by restricting the problem slightly. We seek a CMST subject to the constraint that the number of nodes (rather than the sum of the weights of the nodes) in any subtree not exceed a prespecified maximum. Since our primary intent here is to study the performance of CMST algorithms, this modification would not, in general, have a significant effect. Indeed, if one preferred, a node

of weight K could be replaced by K nodes of weight 1, providing one is willing to allow the original node of weight K to be split among more than one subtree.

To find a partition of the nodes $V = \{v_i | \ i = 1,2,...n\}$ into subtrees, we begin by making n copies of each node corresponding to the possibilities of the node being in any of n possible subtrees. Thus, v_{ij} corresponds to node v_i being in subtree j.

The problem of obtaining an optimal partition of nodes into subtrees can be thought of as one of selecting an optimal subset from the set $E = \{v_{ij} | \ i=1,2,...n; \ j=1,2,...n\}$. Feasible subsets of E, i.e., those corresponding to partitions satisfying the capacity constraint, will contain 1 v_{ij} for each i and at most m v_{ij}'s for each j. If we associate a weight, w_{ij}, with each v_{ij}, the optimal subset of E (hence the optimal partition of V) is defined as the feasible subset of minimum total weight.

An efficient algorithm (see 14, 21, 35) exists for the solution of the problem of finding the optimal subset of E given the values of w_{ij}. The algorithm, which can be thought of as a matroid interesection [1,17,19,20,29] algorithm or alternatively as a series of shortest path problems in appropriately defined graphs, has a worst case running time of order n^3 and in practice has a running time closer to order n^2 (see 35). Unfortunately, the set of weights, w_{ij}, which correspond directly to the "cost" (contribution to the overall length of the CMST) of v_i in subtree j can be specified only when the problem solution is already known. We can, however, define a set of weights, w_{ij}, which have the property that the optimal partition found using these weights will have a value (sum of weights) which is a

lower bound on the length of the optimal CMST. Thus, we can relax the CMST problem to the problem of finding an optimal partition. This, together with generating subproblems by successively restricting either nodes or arcs, gives rise to an optimum CMST algorithm within the branch and bound context.

An appropriate set of weights, w_{ij}, can be defined as follows. Suppose we are given, for each subtree j, the set $\hat{V}_j$ of nodes permitted in the subtree; a procedure for obtaining the $\hat{V}_j$ will be given below. One can then find T_j, the minimum spanning tree on the nodes in $\hat{V}_j \cup \{v_0\}$. Let d_{ij} be the length of the arc connecting v_i to its predecessor in T_j (i.e., d_{ij} is the length of the last arc in the path from v_0 to v_i in T_j). The following theorem, which is proven in (35), allows us to obtain appropriate w_{ij}:

Theorem 2: The weight of the optimal partition using $w_{ij} = d_{ij}$ is a lower bound on the length of the CMST for the same V and M

Furthermore, it is proven in (35) that other similarly defined w_{ij}'s also preserve this lower bound. In particular, suppose T_j contains a path $(v_o,\ldots v_p, v_q,\ldots v_x,\ldots v_y,\ldots)$ as shown in Figure 1. Let S be the set of nodes $\{v_q,\ldots v_k,\ldots v_s\}$ and let w_k be the largest weight of any node in S. Suppose $w_{pj} > w_{kj}$. Define $\Delta = w_{pj} - w_{kj}$. Then the following theorem holds:

Theorem 3: If a set of weights $w_{ij} = d_{ij}$ is modified by transferring weight Δ from w_{pj} to w_{sj} where Δ, v_p and v_s are defined as above, the weight of the optimal partition is still a lower bound on the length of the CMST for the same V and m.

Theorem 3 allows us to transfer weight from a node to its successors in T_j in order to guarantee that the lower bound obtained

from the partitioning problem is at least as tight as the bound obtained using an MST, as is done in (2), (3), and (4). A proof that this can always be done is given in (35). As an example of how this works, consider the network shown in Figure 2a. The MST for this network and the node weights corresponding to it are shown in Figure 2b. These weights correspond to the bound obtained using an MST. Suppose, however, that we restrict v_3 from being part of a given subtree j. The weights shown in Figure 2c would then be obtained if we simply set $w_{ij} = d_{ij}$. Note in particular that w_{2j} has been reduced from 5 to 1. This reduction in w_{2j} could result in a loosening of the lower bound. Theorem 3 allows us to transfer up to $\Delta = w_{1j} - w_{2j}$, i.e., 7 units of weight, from w_{1j} to w_{2j} and obtain the weights shown in Figure 2d. Note that the w_{ij} in Figure 2d are at least as great as the w_{ij} in Figure 2b. Thus, the lower bound obtained using the w_{ij} in Figure 2d will be at least at tight as that obtained using an MST. In fact, the bound so obtained is significantly tighter, as is shown by the computational experience given in Section V.

We now turn to the question of the branching rule within the branch and bound procedure. Little was said by Chandy and Russell, Chandy and Lo, and Elias and Ferguson on the order in which subproblems are considered in the branch and bound procedure. Classically, two approaches are available. The first is to always consider the subproblem with the least lower bound. Alternatively, one can use depth first search, where one always solves most recently generated subproblems before returning to older subproblems. There are advantages and disadvantages to both approaches.

The first approach allows one to proceed without any good feasible solutions to guide the process. The assumption is that subproblems with the lowest lower bounds will give rise to the best feasible solutions. Hence, one prefers to explore these subproblems first in the hope that they will give rise to low cost feasible solutions which will eliminate other subproblems (with higher lower bounds) from consideration. Also, by examining subproblems in this order, one is continually narrowing the range between the upper and lower bounds, and hence, has the option of terminating the algorithm when the interval shrinks to some prespecified width.

There are, however, two major drawbacks to this approach. First, one must keep (a potentially large number of) subproblems around in order to select the next one. Thus, the storage required for the procedure is potentially exponential. In practice, it was storage, not running time, which was the active constraint on problem size in previously developed techniques. One could temporarily store subproblems in secondary storage, but this would complicate and slow down the procedure.

Second, by considering problems in ascending order of lower bound, one will, in general, be sequentially considering dissimilar subproblems. Thus, one cannot easily take advantage of information obtained in the solution of one problem for the solution of another. For example, in the Elias and Ferguson technique, the similarity between modified MST's for related subproblems cannot be easily exploited if this first procedural outline is adopted.

Using depth first search overcomes both of these objections. Indeed, a great deal of simplification is obtainable both in the genera-

tion of subproblems and in obtaining solutions owing to the similarity of successively considered subproblems.

The maximum number of subproblems which need be kept around at any time is bounded by the number of nested specifications it is possible to make. Thus, if one is restricting nodes, the bound is n; if one is including or excluding arcs, the bound is $\binom{n}{2}$. This essentially eliminates storage as an active constraint on the size of the problem which can be considered.

A further reason for using depth first search is that the major reasons one would ordinarily choose the first procedure are not present here. Any of the existing heuristics can be used to quickly generate a good upper bound. Furthermore, the procedures developed in the sequel lend themselves to generating feasible solutions for all subproblems. Thus, a good upper bound is always available.

Hence, depth first search is used in developing the techniques in the sequel. It should be further noted that the philosophy used in developing these techniques was to create the simplest, most flexible framework within which to work so that a variety of acceleration techniques could be developed and tested. The concentration is on restricting the number of subproblems examined (which is exponential in n) rather than the amount of work spent on each subproblem (which is a low order polynomial in n).

III. NODE PARTITIONING

The first exact technique built around the above relaxation is one which generates subproblems by restricting the subtrees a node is allowed into. The procedure is described below. We begin by describing the initialization procedure.

Step 0: (Initialize)

0.1) Find an upper bound, z^*, using a heuristic to generate a good, feasible solution.

0.2) Find an MST, T, and identify arcs (v_0,v_{i_1}), (v_0,v_{i_2}), $\ldots,(v_0,v_{i_k})$.

0.3) Reorder the nodes so that $v_{i_1},v_{i_2},\ldots v_{i_k}$ are now v_1, $v_2,\ldots,v_k$.

0.4) For $1 \leq n$, set $R_{ij} = \begin{cases} \text{TRUE} & i \neq j \\ \text{FALSE} & i = j \end{cases}$

[R_{ij} is a logical variable which is set to TRUE if v_{ij} has been removed from consideration in this subproblem. Observations made above allow us to remove some v_{ij} immendiately].

0.5) For $k < i \leq n$, set $R_{ij} = \begin{cases} \text{TRUE} & i > j \\ \text{FALSE} & i \leq j \end{cases}$

0.6) For $i \leq j \leq n$, find an MST, T_j, on $\hat{V}_j \cup v_o$, where $\hat{V}_j = \{v_i | R_{ij} = \text{FALSE}\}$ i.e., $\hat{V}_j$ is the set of nodes permitted in subtree j.

0.7) Set $w_{ij} = d_{ij}$, where d_{ij} is the distance from v_i to its predecessor in T_j.

0.8) For $j = 1,2,\ldots, k$, exchange weight between w_{ij} for

different values of i so that the resulting modified weights, w'_{ij}, satisfy:

$$w'_{ij} \geq w_{i0}$$

where $w_{i0} = d_{ip_i}$ in the unconstrained MST, T, generated in Step 0.2 above.

0.9) For j = k+1,..., n, exchange weight between w_{ij} so that the resulting w'_{ij} satisfy

$$w'_{ij} \geq w'_{ij} - 1$$

The justification for all of these steps was given in Section 2.

Steps 0.8 and 0.9 guarantee that the individual w'_{ij} will all be at least as great as the weights assigned using unconstrained MST. Hence, this is a realization of the statement that the lower bound obtained using this procedure must be at least as great as the lower bound obtained using an MST. This also holds true for subproblems. Thus, we have initialized a subproblem with nodes 1,2,..., k forced into subtrees 1,2,..., k, respectively, since, for $i \leq k$, R_{ij} = TRUE for $i \neq j$. In the course of the depth first search, we keep track of the following variables:

d = The depth of the search, i.e., the number of nodes which have been forced. d is initialized to k.

d_{MIN} = The minimum allowable depth. d_{MIN} is initialized to k since at least k nodes should will be forced.

IW_d = The subtree which the d^{th} node is forced into.

The depth first search proceeds by forcing node k+1 into subtree 1; i.e., d is set to k+1 and IW_{k+1} is set to 1. It continues either by increasing d (to force another node) changing IW_d (to force

a node into a different subtree) or decreasing d (to release a node after forcing it successively through all subtrees). The depth first search procedure follows.

Depth First Search Procedure

Step 0: Initialize problem (Steps 0.1 through 0.9 above).

Set $d = k+1$

Set $d_{MIN} = k+1$ Set $IW_d = 1$

Step 1: Solve the currently defined subproblem; i.e., find a lower bound z_L, and a feasible solution z_F.

Step 2: If the current subproblem fathoms; i.e., $z_L \geq z_F$, go to Step 3; otherwise go to Step 5.

Step 3: Set $IW_d = IW_d + 1$

If $IW_d > NMAX_d$ go to Step 4; otherwise set up a new subproblem and go to Step 1. $NMAX_d$ is the highest indexed subtree which the node at depth d may be forced into. In section 3 we observed that one should not skip over subtrees. Thus:

$NMAX_d = \max (K, \max [IW_i])$ $K < i < d$

Step 4: Set $d = d+1$

If $d < d_{MIN}$ stop; otherwise set up a new subproblem and go to Step 1.

Step 5: Set $d = d+1$

Set $IW_d = 1$

Set up new subproblem and go to Step 1.

To set up a new subproblem, one need only modify the values of a few R_{ij} to impose (or remove) the restriction implied by the alternation of d and IW_d. Thus, after d is set to d+1 and IW_d is set to 1:

$$R_{dj} = \begin{cases} \text{FALSE } j = 1 \\ \text{TRUE } j > 1 \end{cases}$$

After IW_d is set to $IW_d + 1$:

$$R_{d,IW_d} = \text{FALSE}$$

$$R_{d,IW_d-1} = \text{TRUE}$$

After d is set to d - 1:

$$R_{d+1,j} = \begin{cases} \text{FALSE } j \leq NMAX_d \\ \text{TRUE } j > NMAX_d \end{cases}$$

The search space can be further pared using the corollary to Theorem 1. If, in any subproblem, one finds that two nodes, v_i and v_j, both forced into the same subtree appear in separate subtrees in the MST formed on the set of permissable nodes in that subtree, then the subproblem may be discarded.

As was mentioned, this optimal technique based on generating a partition lends itself simply to obtaining a feasible solution to each subproblem. The partition generated at each step is feasible. One need only generate MST's on each group of nodes to obtain a feasible solution. A simple acceleration technique, which proved to be quite effective in practice, was to reorder the nodes $v_{k+1}, \ldots, v_n$ by distance from v_0, nearest first. This tended to increase the lower bound most rapidly. Such nodes, when restricted to a single subtree, were absent from all others, and the "deprived" subtrees were often forced to connect to v_0 over longer arcs.

The R_{ij} are used in the optimal partitioning procedure in a straight forward fashion; any element v_{ij}, with its corresponding R_{ij} = TRUE, is considered to be removed from the problem.

The weight exchange procedure described as part of the initialization, which guarantees that the weight on each node will be at least as great as its contribution to the length of an MST, is used here as well. At each level, K, in the decision tree, we save the values of the w'_{ij} in a variable referred to as w^K_{ij}. We then demand that $w'_{ij} = w^K_{ij} \geq w^{K-1}_{ij}$ i.e., we exchange weights to enforce the restriction. The justification for doing so is identical to that used in the initialization procedure. Note that this exchange guarantees not only that the lower bound will remain tighter than an MST, but also that the lower bound will be monotone with the depth in the decision tree. Neither of these things is true without the exchange.

IV. ARC RESTRICTIONS

Another method of applying this relaxation technique to the solution to the CMST problem is to restrict arcs; i.e., to force arcs to be either "prohibited" or "required" as was done by Chandy and Russell and Elias Ferguson. Thus, the initialization and subproblem solution are essentially the same as they were in the technique described in the previous chapter, but the method of generating subproblems is different. The solution order is still a depth first search.

Some differences exist in the initialization procedure. Instead of forcing a node into a subtree, we simply "require" the arcs $(i_1,0)$, ..., $(i_K,0)$ which are part of the unconstrained MST. This is, of course, equivalent to what was done in the previous case. It is implemented in a slightly different way, however.

The entire procedure, both during initialization and during subsequent subproblem generation, restricts itself to dealing with established arcs, i.e., arcs which connect a node directly to v_0 or to other nodes connected to v_0 by established arcs. Thus, each "required" arc forces a node into a given subtree and each "prohibited" arc forces a node out of a given subtree. As a new subtree is encountered (i.e...., when an arc of the form (v_0, v_i) is made "required"), we simply assign the next available subtree number to the subtree.

Subproblems are generated by successively restricting (requiring or prohibiting) established arcs. We again use d to represent the depth of the search. Here, however, d refers to the number of forced arcs rather than the number of forced nodes. Note that while

the number of required arcs is limited to n, the number of prohibited arcs is not. We thus have a different type of decision tree than we did in the previous section.

When forcing nodes into subtrees, we dealt with a tree of depth n but with nodes of degree sometimes as great as n. Here, we deal with a binary tree of depth as great as $\binom{n}{2}$. It is not clear, however, especially with the paring techniques being used, which decision tree is actually larger.

The arc chosen for inclusion is,in each case the next arc to be brought in by Prim's MST Algorithm (25); i.e., the shortest arc connecting a node to some node connected to v_0 by required arcs. This has several advantages:

1. The arc chosen, if excluded, will tend to raise the lower bound. This is important as it helps control the size of the decision tree. Since a potentially large number of successive arc exclusions is possible, it is important that an arc exclusion result in an increase of the lower bound as often as possible so that the fathoming process will limit the depth of the search.
2. If the arc is of the form (v_0, v_i), it need only be considered as "required" and not "prohibited." This is a direct consequence of Theorem 1.
3. If the arc (v_i, v_j) is prohibited, and hence, v_i is excluded from the subtree containing v_j, then so are all arcs of the form (v_i, v_K), where v_K is forced into the same subtree as v_j. This is a direct consequence of an Elias and Ferguson result.

We omit the details of the remainder of the implementation of the arc restricting procedure as they are similar to the node restricting procedure described above.

V. COMPUTATIONAL EXPERIENCE

The procedures described in the previous two sections were coded in FORTRAN and run a PDP-10. In this secton, we discuss the results of experiments run to test the behavior of their run time and effectiveness as a function of problem size and constraint tightness.

Problems were generated by reading in n and m and generating random X and Y coordinates for the nodes within a unit square. The location of the center was, in various problems, either random, centered, or in the corner. Euclidean distances were used. Most experiments were run with the center at the geographic center of the unit square; in this way, larger problems could be examined.

Several series of problems were run with identical values of n and m (and, of course, different randomly generated points) to see how stable the running time is from one problem to another. The standard deviation was found to be close to the mean for the problems run. This essentially sa' hat we should not pay close attention to exact run times or the exact number of subproblems examined.

Series of problems were run varying n and m and using both the node restricting and arc restricting procedures. Both procedures were run with the identical problems and, furthermore, the same set of nodes was used (with new nodes added as the problem size grew) for all problems in this series. This results of this experiment are shown in Table 1. As can be seen, the running times for both procedures were comparable and run time grows exponentially with problem size. (It was gratifying to find that the optimal solution values found by both procedures always matched!)

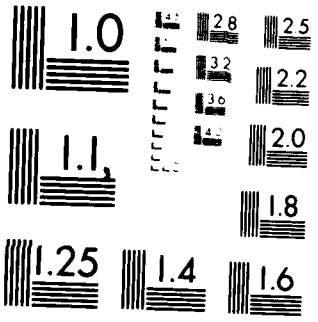
It has already been mentioned that these procedures yield lower bounds which are at least as great as those obtainable using unconstrained MST's. This was verified empirically by actually generating lower bounds with MST's as the algorithm proceeded. The program was to print any exceptions, i.e., any times where the MST generated a higher lower bound; none occurred. Figure 3 shows some typical lower bound values obtained using partitioning and MST's. As can be seen, not only are the partitioning lower bounds greater, but they grow more quickly with depth. This is significant, as a linear increase in lower bound value will reduce the run time exponentially.

To measure the impact of the difference in lower bounds between the MST and partitioning methods, several problems were run first with the partitioning method and then with the MST method of lower bounding. The results of this experiment are shown in Table 2. As can be seen, the partitioning algorithm examines a much smaller number of subproblems, and apparently, its effectiveness increases as the problems grow larger. Thus, although it is somewhat more difficult to evaluate the lower bound using the partitioning algorithm than it is using an MST, it is less than n times as hard to do so. The reduction in the number of subproblems which must be examined appears to be sufficiently great to warrant the use of the partitioning technique. Indeed, as the problem size grows, its attractiveness seems to increase.

AD-A131 674 RESEARCH IN NET MANAGEMENT TECHNIQUES FOR TACTICAL DATA NETWORKS(U) POLYTECHNIC INST OF NEW YORK BROOKLYN R BOORSTYN ET AL. SEP 82 CECOM-80-0579-3 2/2

UNCLASSIFIED DAAK80-80-K-0579 F/G 17/2 NL

END
DATE FILMED
9-83
DTIC



VI. SUMMARY AND CONCLUSIONS

The purpose of this study was to develop improved exact techniques for the solution of the CMST problem (a model of the multidrop line problem) so that known heuristics for the solution to that problem could be examined on a broader class of problems and so that new heuristics could be developed on the basis of what was learned. Much of this happened. An improved exact technique, based upon generating lower bounds using partitioning instead of MST's, was developed and computational experiments were run using it. The bounds yielded by these techniques were tighter than those yields by the MST based techniques, and hence, the number of subproblems which had to be examined in order to obtain a solution was smaller. Indeed, the decrease in the number of subproblems examined more than compensated for the increased effort required for the examination of each subproblem. Thus, the new techniques served their purpose in that they permitted the examination of problems not carefully examined before. In particular, it was possible to examine problems with very tight constraints, although it was not possible to examine problems of substantially greater size than had been previously examined.

Even with the improved technique, the growth of run time with respect to problem size was found to be exponential, albeit of a lower order than previously know exact techniques and of a much lower order than the solution space. Thus, one cannot use the technique for large problems. A number of acceleration techniques were developed and incorporated into the procedure.

Thus, it was possible to examine sufficiently interesting problems using the exact technique to make several insights into the problem. The first is that the performance of the known heuristic degrades as the constraint tightness increases and improves as the problem size increases. An imporved heuristic [33] was also developed on the basis of this study.

REFERENCES

1. Bruno, J., Weinberg, L., "Generalized Networks: Networks Embedded on a Matroid, Part I," NETWORKS, Vol. 6, No. 1, January 1976, pp. 53-94.

2. Chandy, K.M., Lo, T., "The Capacitated Minimum Spanning Tree," NETWORKS, Vol. 3, No. 2, 1973, pp. 173-182.

3. Chandy, K.M., Russell, R.A., "The Design of Multipoint Linkages in a Teleprocessing Tree Network," IEEE TRANS. ON COMPUTERS, Vol. C-21, 1972, pp. 1062-1066.

4. Elias, D., Ferguson, M.J., "Topological Design of Multipoint Teleprocessing Networks," IEEE TRANS. ON COMM., Vol. C-22, 1974, pp. 1753-1761.

5. Esau, L.R., Williams, K.C., "On Teleprocessing System Design, Part II," IBM SYST. J., Vol. 5, No. 3, 1966, pp. 142-147.

6. Fisher, M.J., "Efficiency of Equivalence Algorithms," COMPLEXITY OF COMPUTER COMPUTATIONS, Plenum Press, 1972.

7. Ford, L.R., Fulkerson, D.R., FLOWS IN NETWORKS, Princeton University Press, Princeton, New Jersey, 1962.

8. Frank, H., Chou, W., "Topological Optimization of Computer Networks," PROC. IEEE, Vol. 60, Nov., 1972, pp. 1385-1397.

9. Frank, H., Frisch, I.T., Chou, W., Van Slyke, R., "Optimal Design of Centralized Computer Networks," NETWORKS, Vol. 1, No. 1, 1972, pp. 43-57.

10. Geoffrion, A., PERSPECTIVES ON OPTIMIZATION, Addison-Wesley, Reading, Mass., 1972, pp. 137-162.

11. Johnson, E., "On Shortest Paths and Sorting," PROC> ACM ANNUAL CONF., August, 1972, pp. 510-517.

12. Karnaugh, M., "Multipoint Network Layout Program" International Business Machine Corp., REPORT #RC3723, 1972.

13. Kershenbaum, A., "Computing Capacitated Minimal Spanning Trees Efficently," NETWORKS, Vol. 4, No. 4, October 1974, pp. 299-310.

14. Kershenbaum, A., Boorstyn, R., "Centralized Teleprocessing Network Design," PROC. NATIONAL TELEC. CONF., December 1976.

15. Kershenbaum, A., Chou, W., "A Unified Algorithm for Designing Multidrop Teleprocessing Networks," IEEE TRANS. ON COMM., Vol. C-22, 1974, pp. 1762-1772.

16. Kershenbaum, A., Van Slyke, R., "Computing Minimum Spanning Trees Efficiently," PROC. ACM ANNUAL CONF., August 1972, pp. 518-527.

17. Krogdehl, S., "A Combinatorial Base for Some Optimal Matroid Intersection Algorithms," TECH. REPORT SAN-CS-74-468, Nov. 1974, Computer Science Department, Standford University.

18. Kruskal, J.B., "On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem" PROC. AMER. MATH. COS., Vol. 7, 1956.

19. Lawler, E., "Matroid Intersection Algorithms," MATHEMATICAL PROGRAMMING, Vol. 9, No. 1, 1975.

20. Lawler, E., "Matroids with Parity Conditions: A new Class of Combinatorial Optimization Problems," MEMORANDUM No. ERL-M334, Electronics Research Laboratory, College of Engineering, University of California, Berkeley, Nov. 1971.

21. Lawler, E., COMBINATORIAL OPTIMIZATION: NETWORKS AND MATROIDS, Holt. Reinhart and Winston, 1976.

22. Lin, S., "An Effective Heuristic Algorithm for the Traveling Salesman Problem" OPERATIONS RESEARCH, 1972, pp. 498-516.

23. Little, J.D.C., Murty, K.C., Sweeney, D.W., Karcl, C., "An Algorithm for the Traveling Salesman Problem" OPERATIONS RESEARCH, Vol. 11, November 1976, pp. 972-989.

24. Martin, J., SYSTEM ANALYSIS FOR DATA TRANSMISSION, Englewood Cliffs, New Jersey, Prentice-Hall, 1972.

25. Prim, R.C., "Shortest Connection Networks and Some Generalizations," BELL SYST. TECH. J., Vol. 36, 1957, pp. 1389-1401.

26. Reinfeld, N.V., Vogel, W.R., MATHEMATICAL PROGRAMMING, Printice-Hall, Englewood Cliffs, New Jersey, 1958.

27. Rosenstiehl, P., "L'arbre Minimum d'un Graphe, "THEORY OF GRAPHS, P. Rosenstiehl, Ed., New York: Gordon and Breach, 1967.

28. Sharma, R.L., El-Bardai, M.T., "Suboptimal Communications Network Synthesis," PROC. INTERNATIONAL CONF. ON COMM. June 1970, pp. 19.11-19.16.

29. Tutte, W.T., INTRODUCTION TO THE THEORY OF MATROIDS, American Elseview, 1971.

30. Whitney, H., "On the Abstract Properties of Linear Dependence," AMER. J. MATH., Vol. 56, 1935, pp. 509-533.

31. Whitney, V.K.M., "Comparison of Network Topology Optimization Algorithms," PROC. 1972 ICCC, 1972, pp. 332-337.

32. Papadimitriou, C.H., "The Complexity of the Capacitated Tree Problem," NETWORKS, Vol. 8, No. 3, 1978, pp. 217-230.

33. Kershenbaum, A., Boorstyn, R., and Oppenheim, R. "Second Order Greedy Algorithms for Centralized Teleprocessing Network Design" IEEE TRANS. ON COMM., October 1980, p. 1835-1838.

34. Gavish, B., "New Algorithms for the Capacitated Minimal Directed Tree Problem", Proceedings of ICCC 80, Port Cester, NY, October 1980, p. 996-1000.

35. Kershenbaum, A., "Centralized Teleprocessing Network Design", Ph.D. Thesis, Polytechnic Inst. of NY, 1976.

Table 1: Number of Subproblems Examined

n	m	Node Restricting	Arc Restricting
8	2	16	34
8	3	13	24
10	2	120	199
10	3	57	68
10	4	67	73
12	2	298	696
12	3	375	362
12	4	171	138
14	2	766	723
14	4	526	379
16	3	not run	1085
16	4	818	737
18	3	not run	6832

TABLE 2: COMPARISON OF NUMBER OF SUBPROBLEMS EXAMINED USING MST AND PARTITIONING AS LOWER BOUNDS

n	m	NUMBER OF SUBPROBLEMS (MST)	NUMBER OF SUBPROBLEMS (PARTITIONING)
8	3	87	24
12	3	2,767	362
20	7	4,205	146

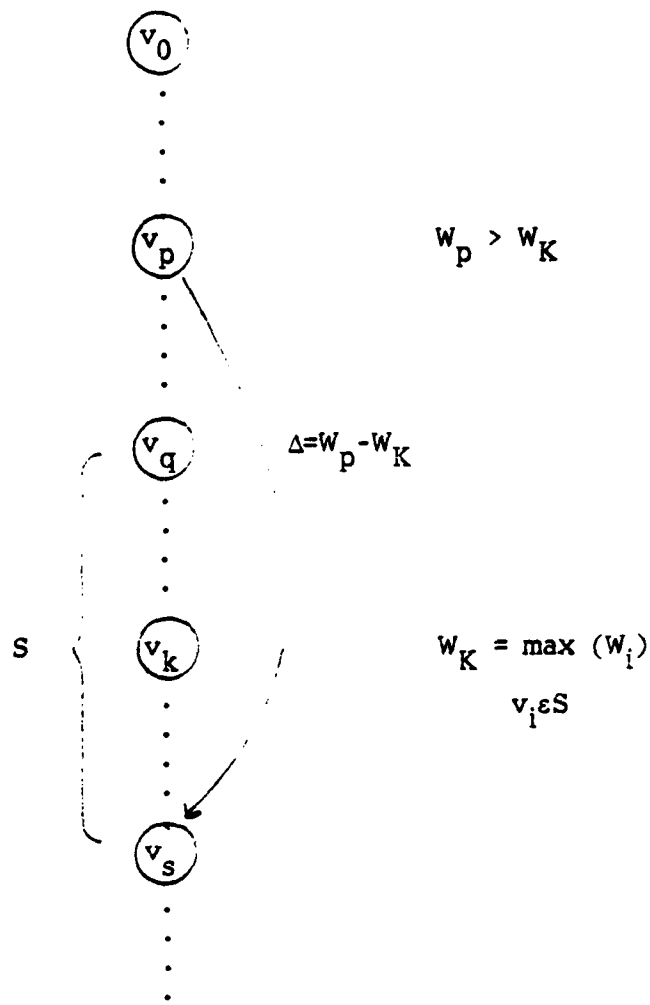
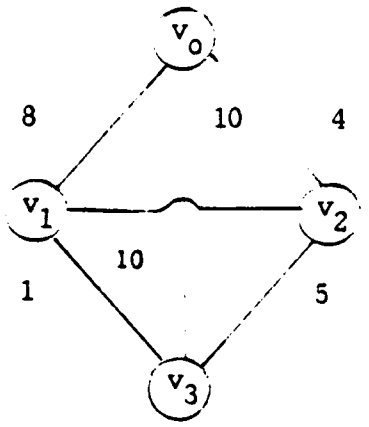
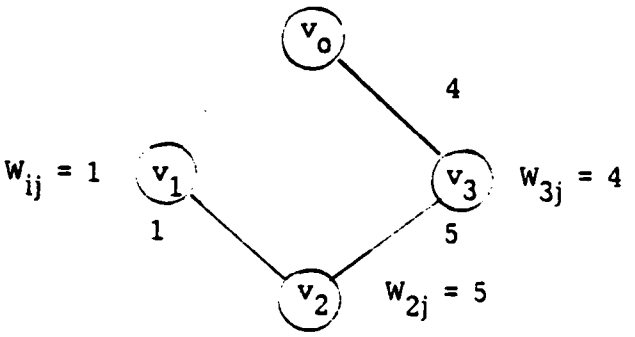


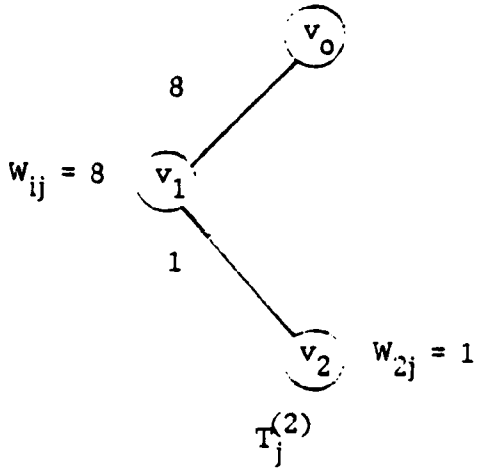
Figure 1



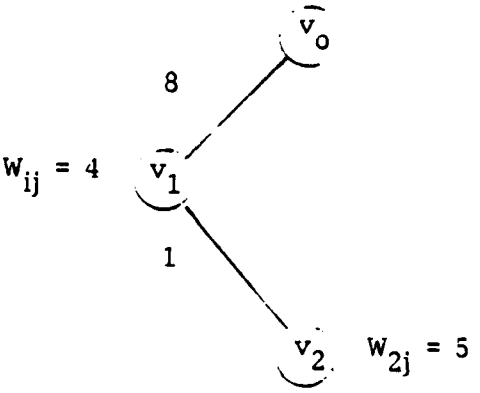
2a



2b



2c



2d

Figure 2

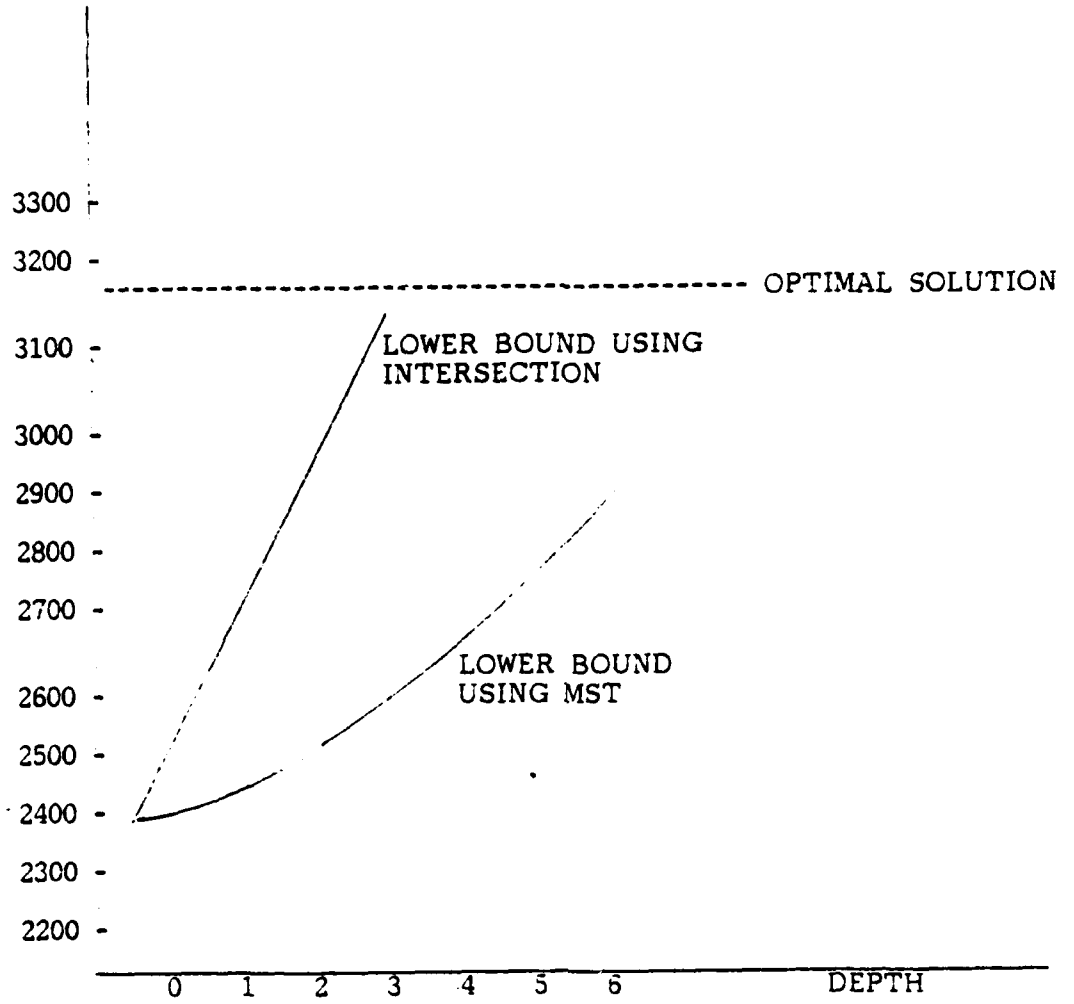


FIGURE 3: COMPARISON OF LOWER BOUND VALUES USING MST AND INTERSECTION

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER CECOM-80-0579-3	2. GOVT ACCESSION NO. AD-A131674	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Research in Net Management Techniques for Tactical Data Networks		5. TYPE OF REPORT & PERIOD COVERED 3rd Semiannual 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Dr. Robert Boorstyn Dr. Aaron Kershenbaum		8. CONTRACT OR GRANT NUMBER(s) DAAK 80-80-K-0579
9. PERFORMING ORGANIZATION NAME AND ADDRESS Polytechnic Institute of New York 333 Jay Street Brooklyn, New York 11201		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 1L1.61102.AH48.DF01
11. CONTROLLING OFFICE NAME AND ADDRESS CDR, CECOM DRSEL-COM-RF-2 Ft. Monmouth, New Jersey 07703		12. REPORT DATE September 1982 13. NUMBER OF PAGES 83
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of th UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for Public Release;
Distribution Unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Packet Radio, Packet Networks, Packet Switching, Network Management, Communications Protocols

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report is the third semi-annual report covering research in Packet Radio communication networks design and analysis. Presented herein is an approach for the throughput analysis of arbitrary multi-hop packet radio networks a teleprocessing network design technique, and a mathematical technique for combinational organization problems.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

CF:
DRSEL-COM-RF
DRSEL-COM-RF-2 (10 Copies)
DRSEL-COM-D